PIPER™

ADVANCE INFORMATION

FEATURES:

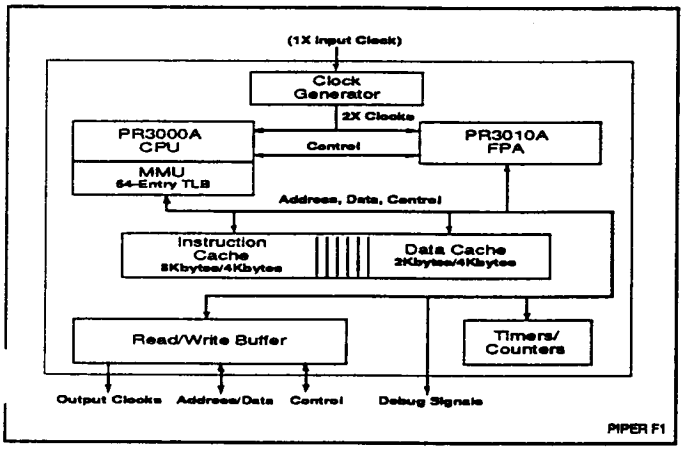
- Highly integrated single-chip RISC processor that contains the following:
  - CPU (PR3000A based)
  - FPA (PR3010A based)
  - Write/Read Buffer
  - Instruction and Data Caches
  - 32-bit Counters/Timers
- Instruction set compatible with the PR3000A RISC CPU and PR3010A FPA (MIPS-1 ISA)
- 32-bit RISC Processor
  - Thirty-two general 32-bit registers
  - Memory Management Unit (MMU) with 64-entry TLB
- Floating Point Accelerator
  - Sixteen 64-bit Floating Point Registers
  - Fully conforms to ANSI/IEEE Std. 754-1985
- Write/ReadBuffer
  - 4-word deep write buffer with read priority
  - 4-word read buffer to support block refill
  - Static column/ page-mode DRAM support
- Available in a 160-pin Metal Quad Flat Pack, at 35, 40, & 45 MHz
- Flexible size instruction and data caches:– Cache sizes can be configured to suit applications needs:
  - 8 Kbytes instruction cache and 2 Kbytes data cache, or
  - 4 Kbytes instruction cache and 4 Kbytes data cache
- Flexible, asynchronous system interface allows simple, low-cost designs
- Intelligent read buffer controller minimizes cache miss penalty for block refills from simple non-interleaved memory systems
  - Dynamic bus sizing feature allows direct interface to 8-bit memories/devices
- All critical high frequency signals isolated in package
  - single input clock (1x clock)
  - two output system clocks
- Two independent 32-bit counters/timers
  - Can be used as watchdog timers, profiling/ scheduling clocks or an event counter.

1.0 DESCRIPTION

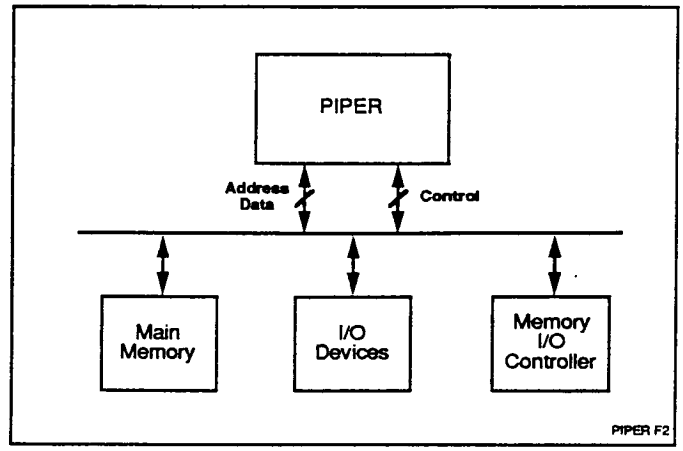
The PIPER (Properly Integrated Performance Easy Rider) is a highly integrated single-chip RISC processor. The PIPER includes a 32-bit RISC CPU, ANSI/IEEE 754-1985 compliant floating point accelerator (FPA), 8Kbytes of instruction cache and 2 Kbytes of data cache (configurable also as 4Kbytes each of instruction and data cache), 4-deep write buffer, 4-deep read buffer and two 32-bit counters/timers. It provides a flexible, asynchronous system interface that allows simple, low-cost designs.

The PIPER's high-integration, high-performance and small footprint makes it ideal for applications such as cost-sensitive laser printer controllers, X-terminal servers and high-performance desk-top workstations.
The "Easy Rider" design isolates the high-speed/frequency critical paths on chip, making system design flexible and simple.

LOGIC DIAGRAM



SYSTEM APPLICATION



PERFORMANCE SEMICONDUCTOR CORPORATION

Means Quality, Service and Speed

4/1/92

The PIPER uses an external multiplexed Address/Data bus to minimize pin count and package size. The PIPER's on-chip clock generation unit and simple initialization modes makes system design simple by eliminating the need for an external delay line and configuration logic. The PIPER also provides external access to internal cache interface signals to assist in system debugging.

The PIPER is manufactured using PACE III (Performance Advanced CMOS Engineered) Technology. PACE III uses 0.6 micron effective channel lengths with two-level metal and epitaxial substrates. In addition to high performance and density, the technology features latch-up protection and single-event-upset protection.

2.0 PIPER OPERATION

The execution engine of the PIPER is based on the PR3000A CPU and PR3010A FPA. The PIPER can be viewed as a single-chip implementation of an PR3000A/PR3010A sub-system that includes a CPU/FPA core, 8/4 Kbytes of instruction cache, 2/4 Kbytes of data cache, 4-deep write buffer, 4-deep read buffer, clock generation unit and two 32-bit timers/counters. This high level of integration dramatically reduces power and board space requirements and reduces total system cost. The operation of the individual functional units that comprise the PIPER are described below.

2.1 CPU/FPA Core

The CPU/FPA core is based on the PR3000A/PR3010A and supports the MIPS-1 ISA (instruction set architecture). Software based on the PIPER instruction set is therefore binary-compatible with the MIPS-1 ISA. The architecture of the CPU is identical to the PR3000A. The 32-bit RISC CPU has a five-stage instruction pipeline, 32 general-purpose registers, an MMU (memory-management unit) consisting of a 64-entry TLB and an autonomous integer multiply/divide unit. The CPU has a 4 Gbyte virtual address space and the addressing modes are identical to those supported by the PR3000A.

The architecture of the FPA is identical to the PR3010A. The FPA operates as a co-processor (CP1) to the CPU and with associated system software, fully conforms to the requirements of the ANSI/IEEE Standard 754-1985. The FPA supports both single and double precision, binary floating-point arithmetic. The FPA has sixteen 64-bit registers and a 32-bit control/status register that provides access to all the IEEE-Standard exception handling capabilities.

For the programmer's model of the CPU/FPA register set, MIPS-1 instruction set description, exception processing and the memory management system see Reference [1]. The processor instruction pipeline is described in Reference [2].

2.2 Instruction/Data Caches

The PIPER has integrated, direct-mapped and physically-tagged instruction and data caches. The on-chip caches can be configured to suit application requirements via a mode signal. Two types of cache configurations are supported: 8 Kbyte I-cache and 2 Kbyte D-cache (asymmetrical caches), and 4 Kbyte I-cache and 4 Kbyte D-cache (symmetrical caches). The performance difference between the symmetrical and asymmetrical caches is application-dependent. In general, the asymmetrical configuration is more suited to embedded controller-type applications (laser printer and X-terminal controllers), while the symmetrical configuration is better suited to workstations and scientific applications.

The instruction *cache line* (number of words with a unique tag) is 4 words (16 bytes) so block refill size is fixed at 4 words. The CPU supports concurrent refill and execution of instructions known as *instruction streaming*. Instruction streaming often permits instruction cache refill to occur transparently.

The data cache uses a write-through cache update policy and has a one word (4 bytes) line. The data block refill size can either be 4 words or one word. See Section 4.0 Initialization, for details on cache configuration.

To facilitate cache flushing and diagnostics, the instruction and data caches can be swapped. For cache swapping operation see Section 7.1.

2.3 Write/Read Buffer

Because the CPU uses a write-through data cache update policy, PIPER includes a 4-deep write buffer that allows the CPU to operate at full speed without stalling during writes. For each CPU write, the write buffer captures the data word and the address including the access type bits and holds them until they can be written out to the main memory. When a write fills the 4-word FIFO, the CPU stalls on the next write attempt until an empty location becomes available in the buffer.

The PIPER uses a *read priority* arbitration scheme for external bus requests. If there are pending writes in the buffer, and there is a cache miss, the PIPER will complete the current write operation and proceed with the cache read before attempting to retire any pending memory writes (provided there are no read conflicts). The read priority scheme has a performance advantage over a write priority scheme where the write buffer is always flushed or emptied before a cache read operation begins. In a read priority scheme like the one used by PIPER, the write buffer is flushed only if there is a read conflict. A read conflict occurs when the read address matches one of the pending write addresses in the buffer. If a read conflict occurs, the write buffer is flushed before the read is performed. For PIPER operation during conflicts, see Section 3.6.

The read buffer is a 4-deep FIFO designed to efficiently support the block refill and streaming modes of the CPU. The buffer helps to match the bandwidth of the external memory system to the bandwidth of the high-speed internal cache bus. Words can be queued up in the read buffer and then released to the CPU to be written into the caches in a burst fashion. This permits the PIPER to operate with maximum efficiency with any type of memory ranging from interleaved page or nibble-mode DRAM system to simple, slower, low-cost memory systems.

Because the cache line is 4 words, instruction cache misses always require a 4-word block refill. During data cache misses, the Dblk input signal determines whether a 4-word block read or a single-word read is performed. The Dblk input is ignored during instruction cache misses.

The PIPER has two operating modes for processing cache block refills: simple block read (Block mode) and burst block read (Burst mode). In Block mode, the PIPER generates the address of every word of a block during block refill transfers. In Burst mode, the PIPER generates only the first address of the block to the memory and then all of the data words in the block are read sequentially. Section 3.2 explains the bus operation during Burst and Block modes. See Section 4.0 for details on enabling Burst or Block mode.

A single-word read is always performed for uncached accesses. During uncached accesses the read buffer functions as a single-word register.

2.4 Clock Generation Unit

The on-chip clock generation unit accepts an external 1x master clock and generates four internal 2x clocks, Clk 2xSys, Clk2xSmp, Clk2xRd and Clk2xPhi required by the CPU.

The clock generation unit includes a PLL (phase-locked loop) mechanism that synchronizes the external clock with an internal master clock. For PLL initialization details, see Section 4.0. The PIPER generates two outputs, $\overline{\text{SysOut}}$ and SysOut, to be used as clocks for external logic. All interface timings are specified with respect to $\overline{\text{SysOut}}$.

2.5 Timer Operation

The PIPER includes two independent 32-bit timers TIMER A and TIMER B that can be used as profiling/scheduling clocks, watchdog timers or an event counter. The block diagram of the timer unit is shown in Figure 2.1. Each timer operates with $\overline{\text{SysOut}}$ and has two registers, the DATA register and CONTROL register. The CONTROL and DATA registers are memory-mapped and physical addresses 0x1FF20000 through 0x1FF2000C are reserved for these registers as shown in Figure 2.2. The mapping can be optionally disabled via an external signal described below. The CONTROL and DATA register formats are shown in Figure 2.3. The DATA register is loaded with initial count value. The CONTROL register bits are also shown in Figure 2.3.

External signals/pins $\overline{\text{TimerInt}}$, $\overline{\text{TimerDisable}}$ and CountBEnable are associated with the timers. The $\overline{\text{TimerInt}}$ output signal is asserted when either one of the timers/counters count value reaches 0. $\overline{\text{TimerInt}}$ must be externally connected to one of the PIPER's $\overline{\text{Int}}$ (5:0) inputs for an interrupt to be recognized by the CPU. If the $\overline{\text{TimerDisable}}$ input is asserted then physical addresses 0x1FF20000 through 0x1FF2000C are no longer reserved for the timer DATA and CONTROL registers, effectively disabling the timers. This feature is useful if the physical addresses 0x1FF20000 through 0x1FF2000C are required for other functions. CountBEnable is used for event counting. If the C (Count enable) bit is set in the CONTROL register, then the count value decrements only when CountBEnable is asserted. The O (Other timer done) bit is useful for checking the status of both timers with a single read operation. See Section 3.10 Timer/Counter timing, for further details.

All accesses (reads/writes) to the timer registers should be 32-bit words (access type is ignored) via uncached address space (KSEG1), otherwise the result of the operation is undefined.

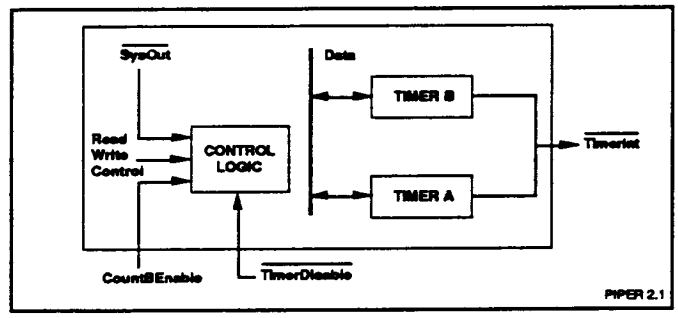
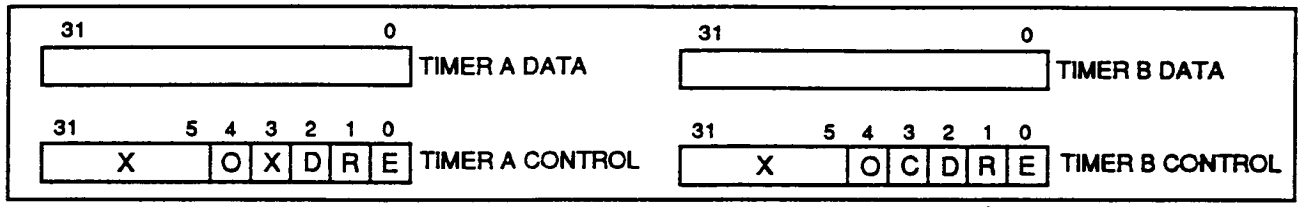


Figure 2.1 Timer Block Diagram

Physical Address	Register
0x 1FF2 0000	TIMER A DATA
0x 1FF2 0004	TIMER A CONTROL
0x 1FF2 0008	TIMER B DATA
0x 1FF2 000C	TIMER B CONTROL

Figure 2.2 Counter/Timer Address Map



Bit	Description
E	Enable timer. When HIGH, timer is enabled and decrements the value loaded in the DATA register at a rate equal to the $\overline{\text{SysOut}}$ clock frequency. When LOW disables timer from counting down. Set LOW during reset.
R	Reload timer. When HIGH, DATA register is reloaded with initial value when the count reaches zero. When LOW, stops counting when the count reaches zero and clears the E (enable) bit.
D	Timer done. When HIGH,count has reached zero. Will be automatically cleared whenever the timer's CONTROL register is read, written or when the other timer's control register is read.
C	Count enable. Only applies to TIMER B. When HIGH, the external CountBEnable signal is used to decrement the timer, instead of $\overline{\text{SysOut}}$. When LOW, external count mode is disabled and TIMER B decrements every cycle. This mode can be used for event counting.
O	Other timer done. Reflects the other timer's D (Timer done) bit.
X	Don't care. Returns zero when read

Figure 2.3 Counter/Timer Register Bit Description

3.0 MEMORY INTERFACE OPERATION

The PIPER features a multiplexed address and data bus with a simple control protocol for ease of design. The bus is designed to interface directly to a system memory controller with a simple handshake. The PIPER protocol for various bus cycles is described below. In each cycle, the address phase (or assertion of LE) indicates the start of a new bus cycle. All timing operations are specified with respect to the PIPER output clock $\overline{\text{SysOut}}$. The term *clock* and *$\overline{\text{SysOut}}$* are interchangeable in the timing descriptions that follow.

A *cycle* is the basic instruction processing unit of the PIPER. Cycles in which forward progress is made, that is, an instruction is retired, are called *run*cycles. An instruction is retired either by its completion, or in the presence of exceptions, its abortion. Cycles in which no further progress is made are called *stall* cycles. Stall cycles are used to resolve conditions like on-chip cache misses, write buffer overflows, FPA interlocks, multiply/divide interlocks etc. All cycles can be classified as either run cycles or stall cycles. *Fixup* stall cycles occur during the final cycle of a stall, that is, the one which occurs prior to re-entering run. The fixup stall cycle is used to re-start the CPU and FPA pipelines and in general to fixup conditions that caused the stall.

All inputs to the PIPER are sampled on the falling edge of $\overline{\text{SysOut}}$ except Dblk, SCpCond (3:2) and $\overline{\text{ByteAssemble}}$ which are sampled on the rising edge of $\overline{\text{SysOut}}$.

3.1 Single-Word Read Cycle

Because of an internal cache miss or an uncached read operation, the PIPER performs a read from external memory, asserts Request to obtain control of the external bus and remains in this state until the bus is granted (for details on bus arbitration, see Section 3.5). When the bus is granted, the PIPER drives the read address and access type on MemBus (31:0), MemAdr (3:2) and asserts LE. To facilitate simple de-multiplexing of the bus, the LE signal is designed to connect directly to the latch enable input of a '373 type latch. The address is always valid for only one clock. MemBus will then become an input for sampling the read data from the memory. The PIPER starts sampling Acknowledge on the falling edge of SysOut in the same cycle as the address is terminated. The PIPER will remain in this state indefinitely sampling Acknowledge on the falling edge of SysOut allowing the memory controller to add wait states until the data is ready. When Acknowledge is asserted, the PIPER samples MemBus(31:0) on the same falling edge that Acknowledge is sampled active. If the operation is a single-word read and ByteAssemble is not asserted, the CPU will resume execution one and a half cycles after the data is sampled. Figure 11.1 shows a single-word, uncached read operation including the bus acquisition time.

Cached and Instr are always LOW for uncached (single-work) reads. The R/W signal is driven HIGH for a read operation.

Figure 11.2 shows a single-word, uncached read with the bus granted; that is, OutEn is asserted prior to the read cycle. The PIPER initiates the read cycle (asserts LE and drives the read address) on the next rising clock edge after Request is asserted. See Section 3.5 for Request and OutEn operation.

3.2 Block Refill Operation

Internal cache misses are serviced using a block or multi-word transfer mechanism known as block refill. Block refill is used to write a multi-word line into the internal caches. The PIPER's instruction block refill size is fixed at 4 words and the data block refill size is 4 words or a single word depending on the Dblk input. Block refills are performed by the PIPER using either Block mode or Burst mode transfers.

3.2.1 Burst Mode Operation

When configured at reset to operate in Burst mode, the PIPER will perform all block refills as bursts except for Byte-assembly (see Section 3.9). A burst operation is similar to the normal single word read cycle with the exception that all word transfers in the block subsequent to the first words, will occur without LEs. Figure 11.3 illustrates burst mode timing where data is available on every clock cycle after the first access is completed. However, any number of wait states may be added between the address phase and Acknowledge. The block address is initially driven on MemBus(31:0) and MemAdr (3:2) with LE. MemAdr(3:2) is equal to zero for the block address. Data is sampled on the same falling clock edge that Acknowledge is sampled asserted. With each Acknowledge, MemAdr(3:2) increments to the address of the next word (without LE) until the whole block is read.

Start is used to reduce the cache miss penalty by minimizing the total number of stalls to service a cache miss. Assertion of Start begins the transfer of data from the on-chip read buffer to the CPU. Optimal assertion of Start allows the CPU to start block refill from the read buffer while the remainder of the block is read into the buffer from external memory. This reduces the number of stall cycles. Start is sampled on the falling edge of the clock and optimally should be asserted 3 cycles before the last Acknowledge so the last word enters the buffer just prior to when it is needed by the CPU. If Start is asserted too early and there are an insufficient number of words in the read buffer, the CPU will stall (using single-word retries) until the rest of the block is read. Single-word retries cause additional stall cycles that increase the cache miss penalty. If Start is not asserted before the last Acknowledge, the PIPER will respond as though Start was asserted with the last Acknowledge. As a result, Start may be tied HIGH at the cost of 3 additional cycles per block read. If Start is tied LOW, the PIPER responds as though Start was asserted with the first Acknowledge, consequently, the remaining words of the block must be supplied on every subsequent clock cycle for maximum performance.

The Cached signal is asserted if the reference is a cached access. This can be used by a memory controller to select between block (cached) or single word (uncached) transfers. The Instr signal is asserted if the reference is an instruction cache miss and is useful if Dlbk is used for data cache refill. Instr is LOW for uncached reads. R/W is driven HIGH for a read operation. Cached, Instr and R/W are asserted with LE and remain valid for the entire Burst or Block mode read operation. See Section 3.5 for Request and OutEn operation during block reads.

3.2.2 Block Mode Operation

If the PIPER is configured to operate in Block mode, all block refills are performed as four consecutive read cycles. The block mode read timing is shown in Figure 11.4. The address for each word in the block is supplied with LE. With each Acknowledge, MemAdr(3:2) increments to the address of the next word just as in the case of Burst mode. For operation of Start, Cached, Instr, R/W, see Burst Mode Operation (Section 3.2.1) and for Request and OutEn operation, see Section 3.5.

3.3 Single-word Refill using Dblk

Dblk is used to select between four-word and single-word refills for data cache miss cycles (Instruction cache block

refill size is fixed at four words and Dblk is ignored during instruction cache misses). During a data cache miss read cycle, the PIPER initially assumes a block transfer (until Dblk is sampled) and therefore drives the block address on the bus. Dblk is sampled on the next rising clock edge after the address phase is terminated. If Dblk is sampled HIGH, a block transfer is signalled. Otherwise, if Dblk is LOW, then a single-word transfer is indicated. If Dblk is HIGH (block transfer), then block refill proceeds either in Burst or Block mode as described in Section 3.2. If Dblk is LOW (single-word transfer), then two cycles later the PIPER drives the word address (actual miss address) or byte address if ByteAssemble was also asserted (see Section 3.9 and Figure 11.5). The memory system controller must ignore the first address and respond only to the second, that is, Acknowledge is only asserted after the second LE. (See Section 3.5 for OutEn operation with Dblk).

3.4 Write Cycles

The PIPER performs bus write cycles to empty or retire data from the on-chip write buffer to external memory. The PIPER asserts Request to obtain control of the external bus and remains in this state until the bus is granted. When OutEn is asserted, the PIPER drives the write address and access type on the MemBus(31:0), MemAdr (3:2) and asserts LE. R/W (LOW), Cached (LOW), Instr (LOW), and Static are also driven with LE. On the next rising clock edge MemBus(31:0) is driven with the write data. While OutEn is asserted, the PIPER continues to drive MemBus(31:0), R/W, Cached, Instr and Static until Acknowledge is received. Figure 11.6 shows the minimum single-word write cycle. Back-to-back writes are shown in Figure 11.7.

If a processor write causes the four-deep buffer to become full, the CPU stalls on the next write attempt until a write buffer location becomes available. (See Section 3.5 for Request and OutEn operation).

3.5 Bus Arbitration

The PIPER's Request and OutEn signals correspond to a bus-request output and a bus-grant input for bus arbitration.

3.5.1 Request (Bus-request) Operations

Request is asserted when the PIPER requires control of the external bus to perform read or write operations. The PIPER remains in this state until the bus is granted. Bus grant is signalled to the PIPER by the assertion of the OutEn input. OutEn is sampled on every falling clock edge after Request assertion. When the bus is granted, the next rising clock edge initiates the address phase of the read or write transaction.

During read or write cycles, if there are no pending writes in the write buffer, Request is de-asserted on the same rising clock edge that terminates the address phase (or the address phase of the last word of the block in Burst or Block mode transfers). If there are pending writes in the buffer, Request remains asserted. If OutEn remains asserted, the address phase of a pending read or write cycle begins on the next rising clock edge following the Acknowledge of a prior cycle.

3.5.2 OutEn (Bus-grant) Operation

If required by the memory system arbiter, a read or write bus cycle may be interrupted any time before Acknowledge is received. This can be useful for resolution of deadlock conditions or other time-sensitive arbitration issues. When PIPER is to relinquish the external bus, OutEn input must be de-asserted on the falling edge of the clock. The PIPER will respond by placing MemBus(31:0), MemAdr(3:2), R/W, Cached, Instr and Static in a high-impedance state on the next rising clock. This allows the memory system controller to grant the bus to another master. When the arbitration conflict is resolved, bus mastership may be returned to the PIPER by re-asserting OutEn. The PIPER will respond by returning its outputs to the same state that they were in prior to the grant being taken away except Static is always de-asserted. This may occur even in the middle of a Block or Burst mode transfer or byte-assembly operation. If the PIPER has begun transferring words to the CPU during block refill, then the CPU will be forced to stall when block refill is interrupted.

The PIPER samples the OutEn input on the falling edge of SysOut. During normal operation OutEn is de-asserted with Acknowledge or any time after. When OutEn is de-asserted MemBus(31:0), MemAdr(3:2), R/W, Cached, Instr and Static become high impedance on the next rising clock edge. If OutEn remains asserted (or is later asserted) after a read or write cycle completes and there are no pending transactions, the PIPER continues to drive MemBus(31:0), MemAdr(3:2), R/W, Cached, Instr and Static. However, the signal values are undefined during this period.

If PIPER is to relinquish the bus during a memory transaction, OutEn may be de-asserted at any time after LE generation. If OutEn is de-asserted before Acknowledge (before a cycle completes) then the PIPER waits for the bus to be re-granted to complete the pre-empted cycle. In this instance, when the bus is re-granted, the address phase is not repeated for the pre-empted cycle. In Burst and Block mode, OutEn may be de-asserted after the address phase as described above. In Burst mode the address phase will not be re-generated when the bus is re-granted. In Block mode, the address phase of the suspended word is not repeated, however the LEs of the remaining words in the block, are generated as in normal operation.

If Dblk is sampled LOW during a data block refill indicating a single-word transfer, the PIPER generates two LEs (address phases) as shown in Figure 11.5. OutEn may be

de-asserted after the first LE (block address) if the bus is to be relinquished. When the bus is re-granted the second LE (word address) of the suspended cycle is generated.

Similar to Dblkoperation, $\overline{\text{OutEn}}$ may be de-asserted after the first LE (word/block address) during byte-assembly. When $\overline{\text{OutEn}}$ is re-asserted, the second LE (first byte address) of the suspended cycle is generated and the cycle continues normally with the LEs of the remaining byte addresses generated.

Figure 11.9 shows an example (two continuous parts) of multiple read/write cycles by the PIPER. If $\overline{\text{OutEn}}$ is kept asserted after a read cycle completes and there is a pending write in the buffer, then the PIPER drives the write address on the next rising edge after the read data is sampled. As the PIPER turns MemBus(31:0) around from an input (read) to an output (write) in half a clock cycle, the read data must be off the bus before PIPER starts driving the address to avoid bus contention problems. One way to avoid bus contention is to de-assert $\overline{\text{OutEn}}$ after a read cycle as shown in Figure 11.9. This technique does not stall the CPU but the write cycle is delayed until the bus is re-acquired.

3.6 Operation During Read Conflicts

All conflicts force the entire write buffer to empty prior to completing the read that caused the conflict. If the CPU is performing a block read, data will be loaded into the read buffer from the memory until the conflict occurs. The write buffer will then be flushed. If the first conflict occurs after data transfer from the read buffer to the CPU has begun, the CPU is forced into a read stall and data transfer to the CPU continues after the conflict is resolved. If the conflict is detected before data transfer to the CPU has started, the CPU remains stalled until the conflict is resolved. Conflicts are resolved transparently to the external system.

If Burst mode is enabled, conflicts are handled differently since the memory block read cannot be interrupted in the Burst mode. In this situation, if a conflict occurs during the block read, the entire block is still read from memory, the write buffer is flushed and the entire block is then re-read from memory. This operation is transparent to the memory system. The PIPER also checks for conflicts during byte-assembly. If a conflict is detected after Dblk is sampled LOW, a write address is generated with the second LE, see Figure 11.5 and the read/miss address is generated after the write buffer is emptied.

3.7 Bus Errors

The occurrence of an extraordinary failure during memory read operations is signalled to the PIPER by asserting $\overline{\text{Berr}}$. $\overline{\text{Berr}}$ is only sampled on every falling clock edge after the address phase of a memory read cycle as shown in Figures 11.1 to 11.4. Assertion of $\overline{\text{Berr}}$ causes the CPU to take a bus error exception after the current read cycle is completed. This means that if $\overline{\text{Berr}}$ is asserted then $\overline{\text{Acknowledge}}$ must be asserted the appropriate number of times to complete the cycle. For example, if $\overline{\text{Berr}}$ is asserted during block refill or byte-assembly, the memory system must continue to supply Acknowledges to complete the block refill or byte-assembly operation. If $\overline{\text{Berr}}$ is asserted during block refill, only the word(s) that caused the error is invalidated before the bus error exception is taken. $\overline{\text{Berr}}$ and $\overline{\text{Acknowledge}}$ may be asserted in the same cycle.

As $\overline{\text{Berr}}$ is not sampled during write cycles, memory failures during writes or writes to illegal addresses may be signalled to the PIPER via the external interrupt inputs.

3.8 Static Column or Page-Mode DRAM Support

The PIPER provides support for DRAM static column, page-mode or nibble-mode operation, improving memory access performance to the same DRAM page. Currently the PIPER supports a minimum DRAM page size of 512 words or 2 Kbytes that is, a 256K-deep DRAM system with 9-bit row and 9-bit column addresses as shown in Figure 3.1. On every memory access, the row and bank addresses are compared with those of the previous access. If the addresses are equal, the $\overline{\text{Static}}$ signal is asserted and may be used by a DRAM controller for generating static column or page-mode accesses.

System simulation and benchmark data show that a large percentage of writes are made to the same DRAM page. Therefore, to improve system performance, $\overline{\text{Static}}$ can be used to retire writes in static column or page-mode.

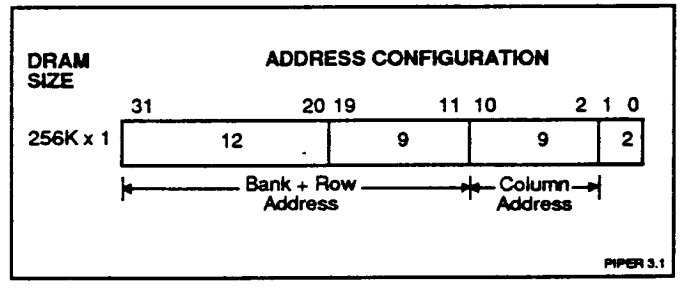


Figure 3.1 $\overline{\text{Static}}$ Address Configuration

If $\overline{\text{OutEn}}$ is de-asserted after a memory cycle, the previous address is disregarded and $\overline{\text{Static}}$ is not asserted for the next memory cycle as shown in Figure 11.7.

3.9 Byte-Assembly (Dynamic Bus Sizing)

To support dynamic bus sizing, PIPER uses the byte-assembly operation to directly perform 32-bit instruction or data reads from 8-bit I/O devices without any intervening glue logic. To save board space, this feature permits the use of a single byte-wide boot PROM instead of four devices. The PIPER does four byte reads from the PROM and assembles the bytes internally to form a 32-bit word.

The $\overline{\text{ByteAssemble}}$ input must be asserted if a memory

read cycle is to be byte-assembled. Both cached and uncached word reads may be byte-assembled. Partial word (half-word and tribyte) reads can also be byte-assembled. ByteAssemble is sampled only on the first rising edge of the clock after the PIPER terminates the address phase of a word read as shown in Figure 11.8. ByteAssemble is not sampled for partial word reads. If ByteAssemble is asserted, then two cycles later the first byte address and access type (based on the Endian) is driven on MemBus (31:0), MemAdr(3:2) with another LE. The memory controller must not respond to the first LE and an Acknowledge should be asserted only for the second LE as shown in Figure 11.8. The data is sampled on MemBus(7:0) on the falling clock edge of Acknowledge. Consecutive byte addresses are driven on the bus until the four bytes of the word are read and internally assembled into a word as shown in Figure 3.2.

Byte-assembly may be used during block refills both in Block and Burst mode. However, ByteAssemble is sampled only once on the first rising clock edge after the address phase is terminated. Therefore if ByteAssemble is asserted then the whole block is read in byte-assembly mode. As shown in Figure 11.8 after the first word is byte-assembled, the first byte address of the second word will be driven on the bus until the whole block is read in Byte-assembly mode. This is true even for Burst mode, that is, the byte address of every word in the block will be driven onto the bus by PIPER identical to Block mode (Burst mode is disabled during byte-assembly).

If ByteAssemble is asserted and the PIPER is configured to operate with individual byte enables (ByteEnc is LOW: see Section 4.2.4), then this mode is disabled and the byte encoding mode is enforced for the access so that the latched Membus(1:0) can be directly connected to an 8-bit I/O device. During byte-assembly, the PIPER defaults to the byte encoding mode and MemBus (3:2) indicates

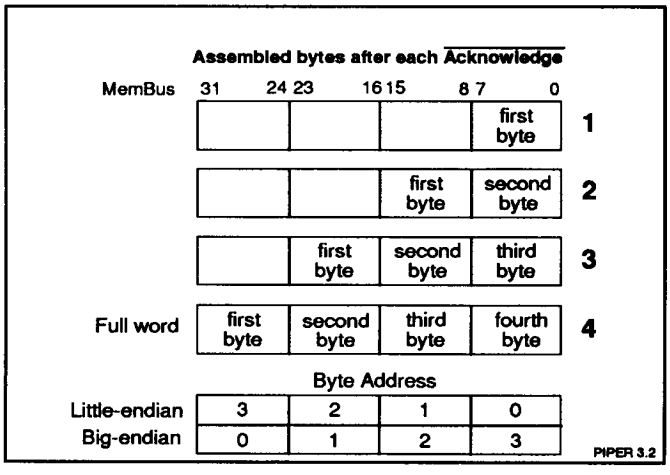


Figure 3.2 Byte-Assembly Operation

byte access (00 = byte access) during the address phase. The byte address is driven on MemBus(1:0), which increments 0, 1, 2, 3 in Big-endian mode or 3, 2, 1, 0 in Little-endian mode, so that the first byte read is assembled on bits (31:24) of the final word as shown in Figure 3.2.

3.10 Timer/Counter Timing

The operation of the timers/counters is shown in Figure 11.10 (three continuous parts). TIMER A DATA is loaded with an initial value 0x11111111 and TIMER A CONTROL is written with 3 (E = 1, R = 1) to enable the re-load mode and start the timer. TIMER B DATA is loaded with a count value of 4 and TIMER B CONTROL is loaded with 0xB (E = 1, R = 1, C = 1) to enable re-load mode, enable external count enable and start timer. TIMER A starts counting (decrements) two cycles after its CONTROL register is written. The count value in the DATA register decrements on the falling edge of the clock.

TIMER B counts (decrements) only when CountBEnable is asserted. CountBEnable is sampled on the falling edge of the clock and if asserted, decrements the count on the next falling clock edge.

TIMER A is loaded with a new count value (value = 6) but the CONTROL register is not re-initialized. The new count value decrements on the next falling clock edge.

TimerInt is asserted when the count value in TIMER B DATA reaches zero (times out). Since TIMER B is programmed to operate in re-load mode, the initial count value (value = 4) is re-loaded in the DATA register and TIMER B continues to operate.

TimerInt may be cleared by reading TIMER A CONTROL, however in Figure 11.10, TIMER A reaches zero on the next clock cycle, so TimerInt remains asserted. TimerInt is cleared by a write to TIMER A CONTROL (E = 1, R = 0). Both the timers are stopped by clearing the CONTROL registers (E = 0). When TIMER A is re-started, the default value in the DATA register (value = 2) is used as the initial count value.

4.0 INITIALIZATION

The reset sequence and the configuration modes of the PIPER are described in this section.

4.1 Reset

The Reset input is used to force processor execution starting at the reset exception vector 0xBFC00000 and initialize processor state. After reset has occurred the following processor state is guaranteed:

A. Status Register State

1. KUc, the current Kernel/User bit, is zero (Kernel mode).
2. IEc, the current interrupt enable bit, is zero (interrupts disabled).

3. TS, the TLB shutdown bit, is zero (TLB enabled).
4. SwC, the swap cache bit, is zero (caches not swapped).
5. BEV, the bootstrap exception vector bit, is one (select bootstrap exception vector).

B. The Random register is set to zero.
C. The on-chip caches and the TLB are in an unknown state and must be initialized via software.
D. The timers/counters are disabled (E bit = 0) and the remaining bits of the CONTROL and DATA register are undefined.

Unlike the PR3000A, the PIPER incorporates special synchronization logic for Reset, so Reset may be asserted and de-asserted from an asynchronous source. It is guaranteed that SysOut and SysOut will be driven after power-up and through the reset sequence.

Three cycles after Reset is asserted, and until it is de-asserted, MemBus(31:0), MemAdr(3:2), Cached, Instr, R/W, and Static are tri-stated, and Request, FPInt, and Timerint are HIGH, and LE is LOW.

To permit the phase-locked loop mechanism to stabilize (see Figure 11.11), Reset must remain asserted for at least 3000 cycles after the power supply and the input clock are stable. The warm reset minimum pulsewidth is also 3000 cycles.

4.2 Configuration Modes

Five mode input signals, BigEndian, BurstRead, CacheSize, ByteEnc and TimerDisable, are used to select the operating configuration of the PIPER. To permit selection of various options, these inputs are tied HIGH or LOW. PIPER operation will be undefined if a mode input signal is dynamically changed during normal operation.

4.2.1 CacheSize

The value of CacheSize input selects between the two cache configurations as shown in Table 4.1. If CacheSize is HIGH, the 4-Kbyte instruction and 4-Kbyte data cache is configured. If CacheSize is LOW then the 8-Kbyte instruction and a 2-Kbyte data cache is configured.

Table 4.1 CacheSize Modes

CacheSize	Configuration
HIGH	4 Kbyte I-cache, 4 Kbyte D-cache
LOW	8 Kbyte I-cache, 2 Kbyte D-cache

PIPER 4

4.2.2 BurstRead

The BurstRead Mode is shown in Table 4.2. When BurstRead is HIGH all block refills are done in Burst mode where only the initial block address is driven by PIPER.

Table 4.2 BurstRead Modes

BurstRead	Operating Mode
HIGH	Burst mode
LOW	Block mode

PIPER 4

When BurstRead is LOW all block refills are done in Block mode where the PIPER drives the address of every word of the block with an LE. For both, MemAdr(3:2) increments with each address phase.

4.2.3 BigEndian

BigEndian selects the byte ordering as shown in Table 4.3. When HIGH, Big-endian byte ordering is selected and when LOW, Little-endian byte ordering is used. PIPER supports dynamic switching between the byte ordering modes via a bit in the CPU Status register. If bit 25 (RE bit) of the Status register is set and bit 1 (KUc bit) is also set, then the CPU will execute code compiled for Endianess opposite to the Endianess for which the system is configured. The value of BigEndian must not change when the byte ordering is switched.

Table 4.3 BigEndian Modes

BigEndian	Operating Mode
HIGH	Big-endian
LOW	Little-endian

PIPER 4

4.2.4 ByteEnc

ByteEnc selects between byte encoding and individual byte enables as shown in Table 4.4. When HIGH, byte encoding is selected and when LOW, individual byte enables are used.

If byte encoding mode is used then MemBus(3:0) encodes the byte selection during address phase (see Figure 4.1). Byte encoding is only valid on MemBus(3:0) during the address phase. At this time, MemBu s(3:2) corresponds to AccTyp(1:0) of the PR3000A CPU (during a read stall or write operation).

Table 4.4 ByteEnc Modes

ByteEnc	Operating Mode
HIGH	Byte encoding mode
LOW	Individual byte enables

PIPER 4

If individual byte enables are used, MemBus(3:0) corresponds to byte enables as shown in Figure 4.2. Byte enables are active HIGH and are valid only during the address phase. Individual byte enables select the same bits on MemBus (31:0) in both Big-endian and Little-endian mode.

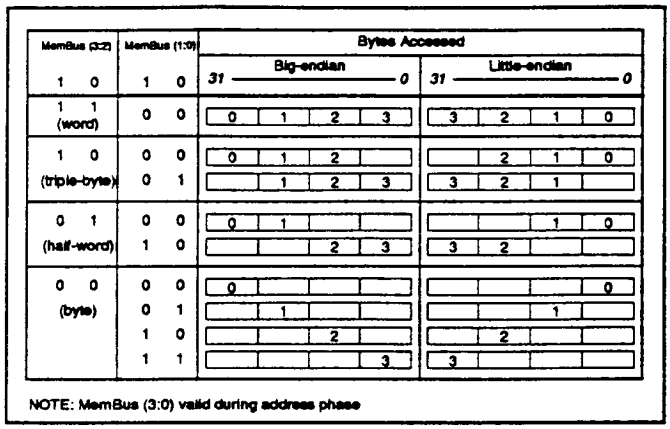


Figure 4.1 PIPER Byte Encoding PIPER 4.1

Byte Enable	Data bits	Byte Address	
		Big-endian	Little-endian
MemBus (3)	Data 31:24	0	3
MemBus (2)	Data 23:16	1	2
MemBus (1)	Data 15:8	2	1
MemBus (0)	Data 7:0	3	0

PIPER 4.2

NOTE: Byte Enables are active HIGH and valid during address phase

Figure 4.2 PIPER Individual Byte Enables

4.2.5 $\overline{\text{TimerDisable}}$

$\overline{\text{TimerDisable}}$ enables or disables the timers/counter as shown in Table 4.5. When HIGH, the timers/counters are enabled and the physical addresses 0x1FF20000 through 0x1FF2000C are reserved for the timers. When LOW, the timers/counters are disabled and the address space 0x1FF20000 through 0x1FF2000C is not reserved.

Table 4.5 $\overline{\text{TimerDisable}}$ Modes

$\overline{\text{TimerDisable}}$	Operating Mode
HIGH	Timers/Counters Enabled
LOW	Timers/Counters Disabled

PIPER 4

5.0 INTERRUPTS

The PIPER has six general purpose interrupt inputs $\overline{\text{Int}}$ (5:0) that are sampled on the falling edge of the clock and are only recognized in run and fixup cycles. The $\overline{\text{Int}}$(5:0) inputs are not sampled in stall cycles. The assertion of any $\overline{\text{Int}}$(5:0) input causes the CPU to abort execution of the current instruction and any following instructions in the pipeline and branch to the general exception vector location 0x80000080. For the PIPER exception processing model see Reference [1].

The interrupt inputs are not latched within the CPU when an interrupt exception occurs but continue to be sampled on the falling edge of the clock and can be read via the CPU Cause register. $\overline{\text{Int}}$(5:0) must be externally synchronized. The source of the interrupt must continue to assert the interrupt input until it is serviced. The Cause register provides a level sensitive indication of the active interrupt or interrupts. The timing of the interrupt inputs is shown in Figure 11.12. The external interrupts may be individually masked or disabled via the Status register.

The FPA signals floating point exceptions to the CPU via one of the interrupt inputs $\overline{\text{Int}}$(5:0). Since $\overline{\text{FPInt}}$ (FPA interrupt) is not internally connected it must be externally connected to one of the interrupt inputs $\overline{\text{Int}}$(5:0). $\overline{\text{FPInt}}$ can be connected directly to any $\overline{\text{Int}}$(5:0) input without intervening synchronization logic.

If the on-chip timer/counter is used in interrupt-driven mode then $\overline{\text{TimerInt}}$ must also be externally connected to one of the interrupt input pins $\overline{\text{Int}}$(5:0). $\overline{\text{TimerInt}}$ can also be directly connected to any $\overline{\text{Int}}$(5:0) without intervening synchronization logic.

6.0 SCpCond (Co-Processor Condition) INPUTS

The CPU supports four co-processor condition inputs CpCond(3:0) that are sampled during run cycles for co-processor branch condition tests. CpCond(0) and CpCond(1) are are used internally and only CpCond(3:2) are available externally as SCpCond(3:2) pins for branch condition inputs. The $\overline{\text{SCpCond}}$(3:2) inputs are internally synchronized with $\overline{\text{SysOut}}$ and can be driven by an asynchronous source. They are sampled on the rising edge of the clock and are recognized only during run cycles as shown in Figure 11.12. If the processor executes a co-processor 2 or co-processor 3 branch condition instruction, the state of the appropriate SCpCond input determines the direction of the branch.

To execute co-processor branch condition instructions, the corresponding co-processor usability bits (bits 28 to 31) in the CPU Status register, must be enabled.

CpCond(1) is used by the FPA (co-processor 1). The FpCond output from the FPA is internally connected to the CPU's CpCond(1) input and can be used for co-processor 1 branch condition tests.

CpCond(0) can be used to determine if the write buffer is empty. The CPU's CpCond(0) is driven during run cycles by an internal signal that is asserted HIGH whenever there are pending writes in the write-buffer, that is, CpCond(0) is driven HIGH whenever the write buffer is not empty. Software could poll the CpCond(0) input using co-processor 0 branch condition instructions to check if the write buffer is empty. CpCond(0) will remain HIGH until $\overline{\text{Acknowledge}}$ is asserted for the last word in the write buffer. To ensure that the timing requirement is met, there

must be at least two instructions between a store and a subsequent CpCond(0) branch test.

7.0 DEBUG SUPPORT

The PIPER's internal cache interface signals including the cache address and tag bus are externally available to help in system debug, program trace and for observability purposes. For a description of these observation signals and timing, see Section 9.0 and Reference [2].

The observation signals are enabled by asserting the $\overline{\text{ObsEn}}$ signal. If $\overline{\text{ObsEn}}$ is de-asserted, all the observation pins are in tri-state. To reduce noise and power, $\overline{\text{ObsEn}}$ should be de-asserted when not in use.

The $\overline{\text{CacheInv}}$ signal is useful for dynamically forcing cached accesses to be uncached. Asserting $\overline{\text{CacheInv}}$ negates the internal cache entry valid bit (TagV) bit read by the CPU and forces a cache miss. This results in a bus read cycle after a current write cycle (if any), completes. Asserting $\overline{\text{CacheInv}}$ does not invalidate a cache entry, that is, the valid bit in the cache line is not negated.

7.1 Cache Swapping

The CPU permits the data and instruction caches to be swapped to facilitate cache flushing and diagnostics. Cache swapping is accomplished via the SwC bit of the Status register. See Reference [1]. When the caches are swapped, the cache that was functioning as the instruction cache becomes the data cache and vice versa.

All memory references within the immediate vicinity of the swap operation must be uncached. Because of the four word instruction cache line, when the caches are swapped, the data cache line becomes four words. Consequently, the swapped mode is acceptable for cache flushing or diagnostics but not for normal operation. Dblk must remain HIGH when the caches are swapped.

8.0 REFERENCES

[1] *MIPS R2000/R3000 RISC Architecture* by Gerry Kane, Prentice Hall, 1987.
[2] *R3000A Processor Interface*, March 9, 1990.

9.0 PIPER SIGNAL DESCRIPTIONS (Functional)

ClockIn	I	Master input clock (1x Clock)
$\overline{\text{Reset}}$	I	Asynchronous reset
MemBus(31:0)	I/O	Memory Bus. A multiplexed address and data bus used to transfer data between the memory system and the PIPER. During the address cycle the address is driven, except on bits (3:2) that have the size of the data transfer (access type). High impedance when $\overline{\text{OutEn}}$ is HIGH.
MemAdr(3:2)	O	Address bits (3:2) of the current bus cycle. Drives the current word address during block refill, if $\overline{\text{Burst}}$ mode is enabled. High impedance when $\overline{\text{OutEn}}$ is HIGH.
Instr	O	When HIGH, indicates that the current bus cycle is cached instruction read. When LOW, indicates a cached data read. Is LOW during uncached reads or write cycles. High impedance when $\overline{\text{OutEn}}$ is HIGH. Asserted with LE valid for the entire cycle.
Cached	O	When HIGH, indicates that the current read bus cycle is cached access. When LOW, the read is a non-cached access. Cached is LOW during write bus cycle. Asserted with LE valid for the entire bus cycle. High-impedance when $\overline{\text{OutEn}}$ is HIGH.
LE	O	Address latch enable. Indicates that an address is currently available on the MemBus. It should be used by an external latch to latch the address off of the MemBus.
$\overline{\text{Request}}$	O	Memory bus request. When LOW, indicates that the PIPER requires access to the MemBus either to transfer data from the write buffer to the memory or from the memory to the read buffer.
$\overline{\text{Acknowledge}}$	I	Memory bus acknowledge. When LOW, it indicates the completion of a memory bus cycle. If the cycle is a read, indicates to the PIPER that the data is valid on the MemBus. If the cycle is a write, indicates that the memory system has accepted the data off of the MemBus.
$\overline{\text{OutEn}}$	I	Memory bus output enable. When LOW, indicates to the PIPER that MemBus access has been granted (in response to $\overline{\text{Request}}$) and that it may begin driving the MemBus and its controls.
R/$\overline{\text{W}}$	O	Memory bus read / write control. When HIGH, indicates to the memory that the current MemBus transaction is a read. When LOW, indicates that the current MemBus transaction is a write. (Asserted with LE and valid for the entire bus cycle.) High-impedance when $\overline{\text{OutEn}}$ is HIGH.
$\overline{\text{Static}}$	O	Static column address. When LOW, indicates that successive memory accesses have the same row address. High-impedance when $\overline{\text{OutEn}}$ is HIGH.
SCpCond(3:2)	I	Coprocessor branch condition inputs internally synchronized with $\overline{\text{SysOut}}$.
$\overline{\text{Int}}$(5:0)	I	External interrupts to CPU. Must be externally synchronized.
$\overline{\text{Start}}$	I	The $\overline{\text{Start}}$ signal input informs the read buffer (during block read) to begin word transfers to the CPU (can be tied HIGH or LOW in special circumstances).
$\overline{\text{ByteAssemble}}$	I	Enable byte-wide read.
$\overline{\text{CacheInv}}$	I	Cache invalidate. When LOW, a cache miss stall is forced for each cached memory cycle.
$\overline{\text{SysOut}}$	O	Inverted system clock.
SysOut	O	System clock.
$\overline{\text{BErr}}$	I	Bus error. Signals the occurrence of a bus error during memory reads.

$\overline{\text{FPInt}}$	O	Floating-point interrupt. Must be externally connected to one of the $\overline{\text{Int}}$ (5:0) inputs.
$\overline{\text{TimerInt}}$	O	Timer interrupt. When LOW, one of the timer/counters has reached zero. Must be externally connected to one of the $\overline{\text{Int}}$ (5:0) inputs if the timer/counters are enabled.
CountBEnable	I	Counter/Timer B Count enable.
$\overline{\text{TimerDisable}}$	I	$\overline{\text{TimerDisable}}$ Mode Input: When LOW, Counters/Timers are disabled; when HIGH, counters/timers enabled.
Dblk	I	Data Block Input: When HIGH, indicates data block refill (4 words) for a data cache miss. When LOW, it indicates a single-word data transfer.
BigEndian	I	BigEndian Mode Input: When HIGH, Big Endian Ordering Mode is active. When LOW, Little Endian Ordering Mode is active.
BurstRead	I	BurstRead Mode Input: When HIGH, Burst Mode will be enabled during block refills. When LOW Block Mode is enabled.
CacheSize	I	CacheSize Mode Input: When HIGH, 4Kbyte (single-word line) data cache and 4Kbyte (4-word line) instruction cache are configured. When LOW, 2Kbyte (single-word line) data cache and 8Kbyte (4-word line) instruction cache are configured.
ByteEnc	I	Byte Encode Mode Input: When HIGH, MemBus (3:0) during address phase, encodes byte selection. When LOW, these signals become individual byte enables.
Resvd (1:0)	NC	Reserved for future use. Leave unconnected to ensure compatibility with future versions.

PIPER SIGNAL DESCRIPTIONS (Observation & Debug Support)

Tag (31:12)	O	A 20-bit bus for transfering cache tags and high addresses between the processor, caches and the read/write buffer.
AdrLo (11:2)	O	A 10-bit bus for transferring low addresses from processor to caches & read/write buffer.
$\overline{\text{Xen}}$	O	Read Enable for the Read Buffer.
$\overline{\text{ObsEn}}$	I	Output Enable for Observation Pins.
$\overline{\text{DRd}}$	O	Data Cache Read Enable.
$\overline{\text{IRd}}$	O	Instruction Cache Read Enable.
AccTyp (2:0)	O	A 3-bit bus used to indicate the size of data being transferred on the internal data bus , whether or not data transfer is occuring, and the purpose of the transfer.
$\overline{\text{Run}}$	O	Indicates whether the processor is in Run or Stall Mode.
$\overline{\text{Exception}}$	O	Indicates exception related information.
FPBusy	O	Signal from the FPA to the CPU indicating a request for a coprocessor busy stall.
$\overline{\text{MemRd}}$	O	Signals the occurrence of a main memory read.
$\overline{\text{MemWr}}$	O	Signals the occurrence of a main memory write.
RdBusy	O	Read buffer busy.

10.0 ELECTRICAL SPECIFICATIONS, COMMERCIAL TEMPERATURE RANGE (T=0° TO 70°C, V=5V ±5%)

10.1 MAXIMUM RATINGS[3]

Symbol	Parameter	Conditions	Min.	Max.	Units
V_{CC}	Supply Voltage		-0.5	+7.0	V
V_{IN}	Input Voltage[1,2]		-0.5	+7.0	V

Notes:
1. V_{IN} Min. = -3.0V for pulse width less than 15ns.
2. $V_{in} \leq V_{CC} + 0.5$
3. Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

10.2 RECOMMENDED OPERATING CONDITIONS [1, 2 & 3]

Grade	Ambient Temperature	GND	V_{CC}
Commercial	0°C to +70°C	0V	5.0V ± 5%

Notes:
1. The case temperature must be limited by using adequate air flow and/or an appropriate heat sink or other thermal management design.
2. The maximum operating junction temperature should be limited to 125°C.
3. For optimum performance and improved reliability, it is recommended that the operating junction temperature should be kept below 85°C.

10.3 CAPACITIVE LOAD DERATING FACTOR

Symbol	Parameter	Conditions	35MHz		40MHz		45MHz		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
C_{LD}	Load Derate		0.5	1	0.5	1	0.5	1	ns/25pF

10.4 DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Conditions	35MHz		40MHz		45MHz		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}	Output HIGH Voltage	V_{CC} = Min. I_{OH} = -4mA	2.4		2.4		2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min. I_{OL} = 4mA		0.4		0.4		0.4	V
V_{IH}	Input HIGH Voltage		2	V_{CC}+0.5	2	V_{CC}+0.5	2	V_{CC}+0.5	V
V_{IL}	Input LOW Voltage (1)		-0.5	0.8	-0.5	0.8	-0.5	0.8	V
V_{IHS}	Input HIGH Voltage (2)		3.0	V_{CC}+0.5	3.0	V_{CC}+0.5	3.0	V_{CC}+0.5	V
V_{ILS}	Input LOW Voltage (2)		-0.5	0.6	-0.5	0.6	-0.5	0.6	V
C_{IN}	Input Capacitance			10		10		10	pF
C_{OUT}	Output Capacitance			10		10		10	pF
I_{CC}	Operating Current	V_{CC} = 5.25V		1100		1300		1500	mA
I_{IL}	Input LOW Current	V_{IN} = GND V_{CC} = Max.	-10		-30		-50		µA
I_{IH}	Input HIGH Current	V_{IN} = V_{CC} V_{CC} = Max.		10		30		50	µA
I_{OZL}	Output 3 State Current LOW	V_{OUT} = 0.5V V_{CC} = Max.	-50		-70		-100		µA
I_{OZH}	Output 3 State Current HIGH	V_{OUT} = 2.4V V_{CC} = Max.		50		70		100	µA

Notes:
1. Transient inputs with V_{IL} and I_{IL} not more negative than –3.0V and –100mA, respectively are permissible for pulse widths up to 15ns.
2. V_{IHS} and V_{ILS} apply to ClockIn.

10.5 AC ELECTRICAL CHARACTERISTICS COMMERCIAL TEMPERATURE RANGE[1&2] (T = 0° to 70°C, V = 5V±5%)

Symbol	Parameter Description	35MHz		40MHz		45MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t_1	Setup time to Clock edge	2		2		1		ns
t_2	Hold time from Clock edge	2		2		1		ns
t_3	$\overline{\text{Request}}$ valid from Clock rising		10		8		6	ns
t_4	Tri-state time from Clock rising		6		5		4	ns
t_5	R/$\overline{W}$ valid from Clock rising		8		7		6	ns
t_6	MemBus (31:0) valid fromClock rising		12		10		8	ns
t_7	LE valid from Clock rising		6		5		4	ns
t_8	LE valid from Clock falling		6		5		4	ns
t_9	Cached valid from Clock rising		12		10		8	ns
t_{10}	Instr valid from Clock rising		12		10		8	ns
t_{11}	$\overline{\text{Static}}$ valid from Clock rising		14		12		10	ns
t_{12}	MemBus (31:0) driven (Low-Z) from Clock rising	3		2		1		ns
t_{13}	$\overline{\text{TimerInt}}$ valid from Clock falling							ns
t_{14}	$\overline{\text{Int}}$ (5:0) to Clock falling setup	5		4		3.5		ns
t_{15}	$\overline{\text{Int}}$ (5:0) from Clock falling hold	0		0		0		ns
t_{16}	$\overline{\text{Reset}}$ pulse width	3000		3000		3000		cyc
t_{17}	LE pulse width	14		12		10		ns
t_{18}	MemBus address to LE falling setup	8		7		6		ns
t_{19}	SCpCond (3:2) to Clock rising setup	2		2		1		ns
t_{20}	SCpCond (3:2) to Clock rising hold	2		2		1		ns
t_{21}	ClockIn HIGH (transition ≤ 5.0ns)	10		8		6		ns
t_{22}	ClockIn LOW (transition ≤ 5.0ns)	10		8		6		ns
t_{23}	ClockIn period	28		25		22		ns
t_{24}*	Output clock HIGH							
t_{25}*	Output clock LOW							
t_{26}	Output clock period							
t_{27}*	$\overline{\text{SysOut}}$ falling to SysOut rising							

Notes:
1. All output times are given assuming 25pF of capacitive load.
2. All timings referenced to 1.5V.
* Valid only after t_{28} is met

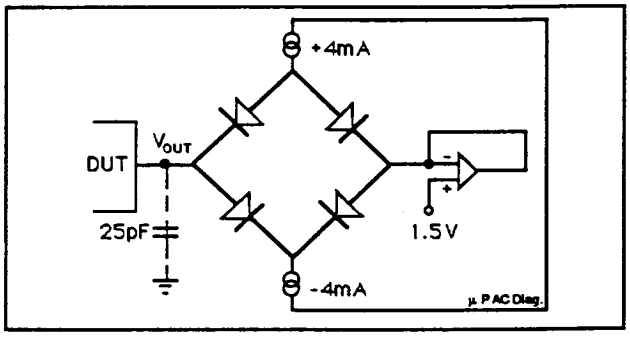


Figure 10.6 Output Loading for AC Testing

10.6 ELECTRICAL SPECIFICATIONS, MILITARY TEMPERATURE RANGE (T= –55° TO +125°C, V=5V ±10%)

10.7 MAXIMUM RATINGS[3]

Symbol	Parameter	Conditions	Min.	Max.	Units
V_{CC}	Supply Voltage		-0.5	+7.0	V
V_{IN}	Input Voltage[1,2]		-0.5	+7.0	V

Notes:
1. V_{IN} Min. = -3.0V for pulse width less than 15ns.
2. $V_{in} \leq V_{CC} + 0.5$
3. Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

10.8 RECOMMENDED OPERATING CONDITIONS [1,2&3]

Grade	Case Temperature	GND	V_{CC}
Commercial	–55°C to +125°C	0V	5.0V ± 10%

Notes:
1. The case temperature must be limited by using adequate air flow and/or an appropriate heat sink or other thermal management design.
2. The maximum operating junction temperature should be limited to 125°C.
3. For optimum performance and improved reliability, it is recommended that the operating junction temperature should be kept below 85°C.

10.9 CAPACITIVE LOAD DERATING FACTOR

Symbol	Parameter	Conditions	25MHz		35MHz		Units
			Min.	Max.	Min.	Max.	
C_{LD}	Load Derate		0.5	1	0.5	1	ns/25pF

10.10 DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Conditions	25MHz		35MHz		Units
			Min.	Max.	Min.	Max.	
V_{OH}	Output HIGH Voltage	V_{CC} = Min. I_{OH} = -4mA	2.4		2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min. I_{OL} = 4mA		0.4		0.4	V
V_{IH}	Input HIGH Voltage		2	V_{CC}+0.5	2	V_{CC}+0.5	V
V_{IL}	Input LOW Voltage[1]		-0.5	0.8	-0.5	0.8	V
V_{IHS}	Input HIGH Voltage[2]		3.0	V_{CC}+0.5	3.0	V_{CC}+0.5	V
V_{ILS}	Input LOW Voltage[2]		-0.5	0.6	-0.5	0.6	V
C_{IN}	Input Capacitance			10		10	pF
C_{OUT}	Output Capacitance			10		10	pF
I_{CC}	Operating Current	V_{CC} = 5.25V		900		1200	mA
I_{IL}	Input LOW Current	V_{IN} = GND V_{CC} = Max.	-10		-30		µA
I_{IH}	Input HIGH Current	V_{IN} = V_{CC} V_{CC} = Max.		10		30	µA
I_{OZL}	Output 3 State Current LOW	V_{OUT} = 0.5V V_{CC} = Max.	-50		-70		µA
I_{OZH}	Output 3 State Current HIGH	V_{OUT} = 2.4V V_{CC} = Max.		50		70	µA

Notes:
1. Transient inputs with V_{IL} and I_{IL} not more negative than –3.0V and –100mA, respectively are permissible for pulse widths up to 15ns.
2. V_{IHS} and V_{LS} apply to ClockIn.

10.11 AC ELECTRICAL CHARACTERISTICS MILITARY TEMPERATURE RANGE[1&2] (T = -55° TO +125°C, V= 5V±10%)

Symbol	Parameter Description	25MHz		35MHz		Unit
		Min.	Max.	Min.	Max.	
t_1	Setup time to Clock edge	4		2		ns
t_2	Hold time from Clock edge	4		2		ns
t_3	$\overline{\text{Request}}$ valid from Clock rising		14		10	ns
t_4	Tri-state time from Clock rising		8		6	ns
t_5	R/$\overline{\text{W}}$ valid from Clock rising		10		8	ns
t_6	MemBus (31:0) valid fromClock rising		16		12	ns
t_7	LE valid from Clock rising		8		6	ns
t_8	LE valid from Clock falling		8		6	ns
t_9	Cached valid from Clock rising		16		12	ns
t_{10}	Instr valid from Clock rising		16		12	ns
t_{11}	$\overline{\text{Static}}$ valid from Clock rising		18		14	ns
t_{12}	MemBus (31:0) driven (Low-Z) from Clock rising	5		3		ns
t_{13}	$\overline{\text{TimerInt}}$ valid from Clock falling					ns
t_{14}	$\overline{\text{Int}}$ (5:0) to Clock falling setup	7		5		ns
t_{15}	$\overline{\text{Int}}$ (5:0) from Clock falling hold	0		0		ns
t_{16}	$\overline{\text{Reset}}$ pulse width	3000		3000		cyc
t_{17}	LE pulse width	18		14		ns
t_{18}	MemBus address to LE falling setup	10		8		ns
t_{19}	SCpCond (3:2) to Clock rising setup	4		2		ns
t_{20}	SCpCond (3:2) to Clock rising hold	4		2		ns
t_{21}	ClockIn HIGH (transition ≤ 5.0ns)	13		10		ns
t_{22}	ClockIn LOW (transition ≤ 5.0ns)	13		10		ns
t_{23}	ClockIn period	40		28		ns
t_{24}*	Output clock HIGH					
t_{25}*	Output clock LOW					
t_{26}	Output clock period					
t_{27}*	$\overline{\text{SysOut}}$ falling to SysOut rising					

Notes:
1. All output times are given assuming 25pF of capacitive load.
2. All timings referenced to 1.5V.
* Valid only after t_{28} is met

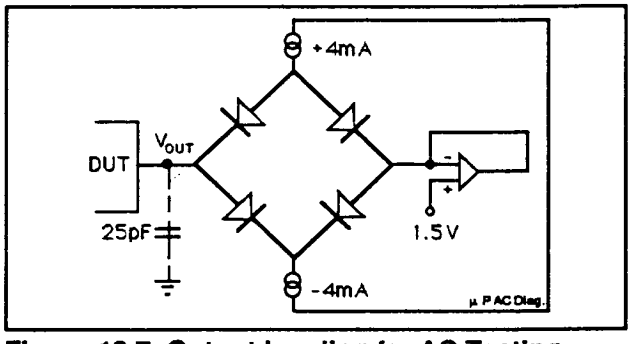
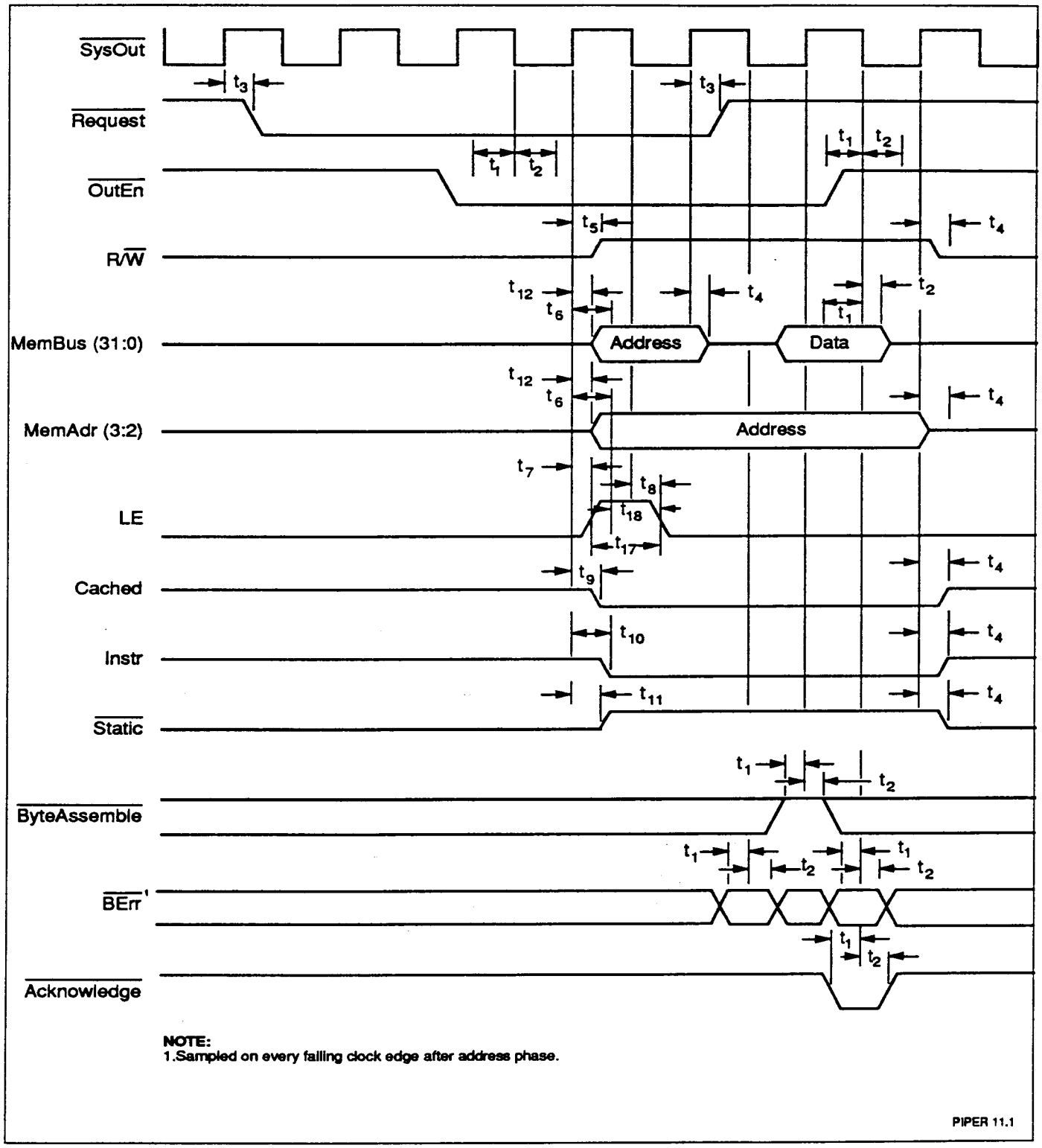


Figure 10.7 Output Loading for AC Testing

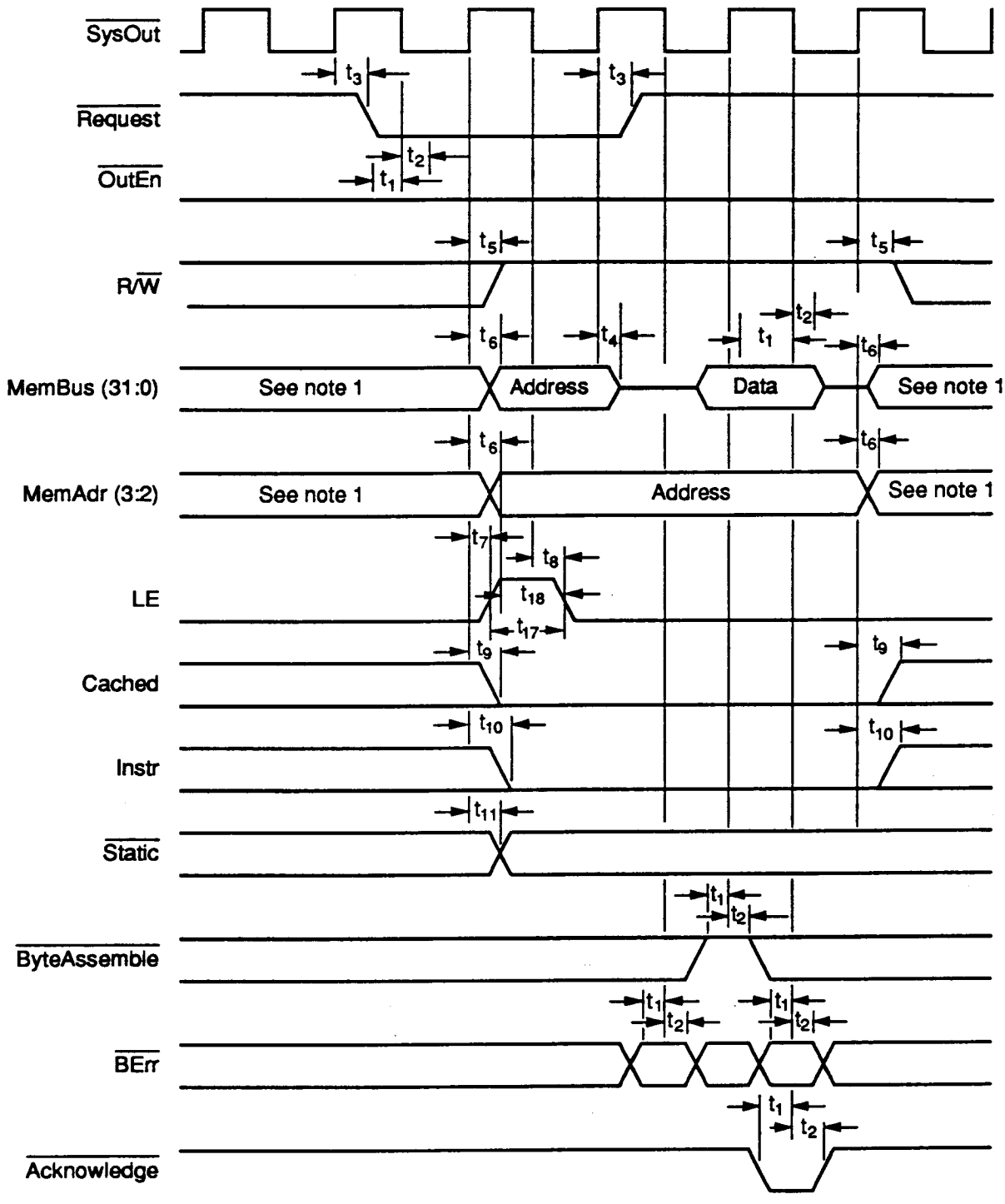
11.0 TIMING DIAGRAMS



NOTE:
1.Sampled on every falling clock edge after address phase.

PIPER 11.1

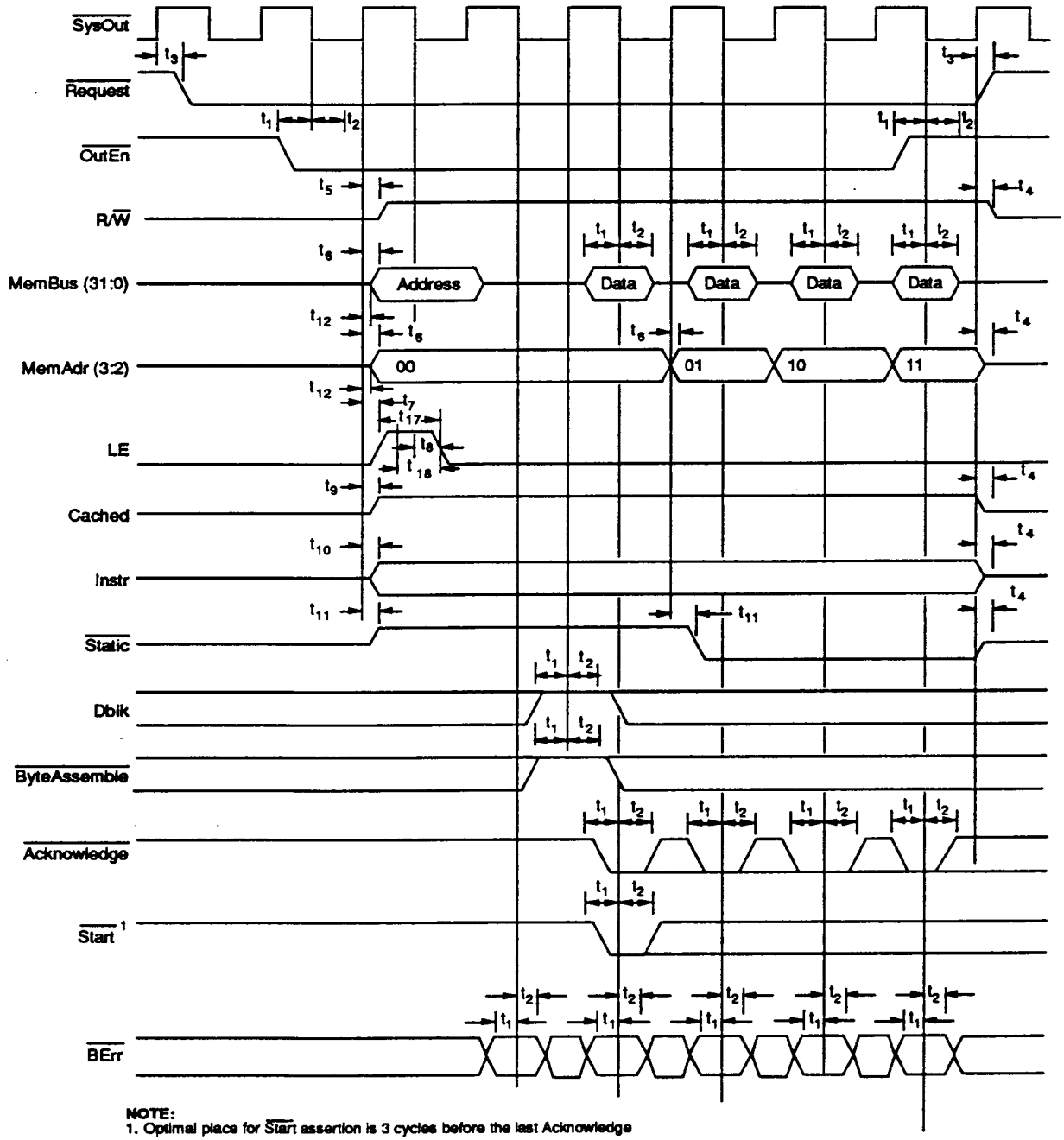
Figure 11.1 PIPER Uncached Single-Word Read with Bus Acquisition (without byte-assembly)



NOTE:
1. Driven by PIPER, with undefined value.

PIPER 11.2

Figure 11.2 PIPER Uncached Single-Word Read with Bus Granted (without byte-assembly)



NOTE:
1. Optimal place for $\overline{\text{Start}}$ assertion is 3 cycles before the last Acknowledge

PIPER 11.3

Figure 11.3 PIPER Burst Read with Bus Acquisition (without byte-assembly) & one cycle delay between words.

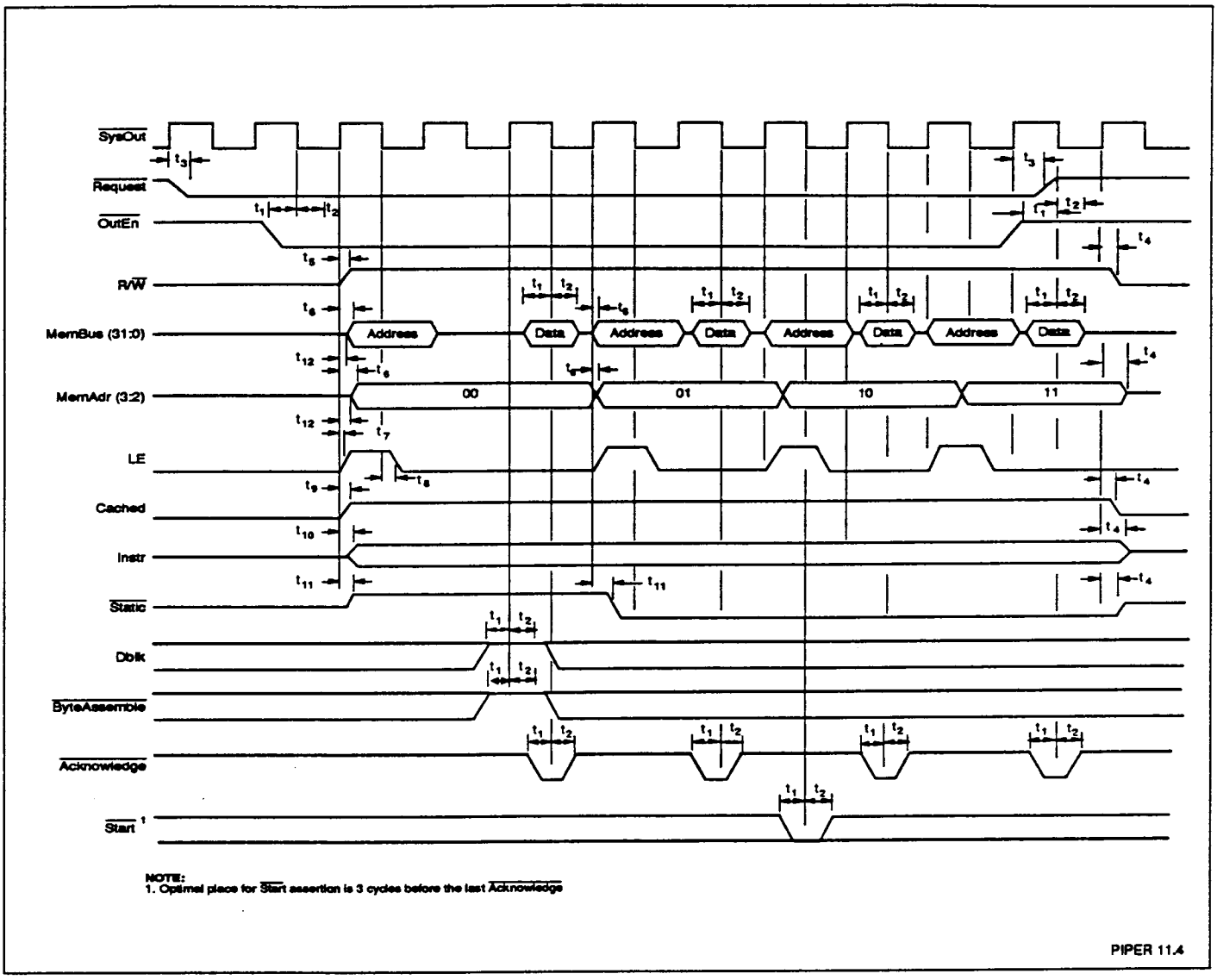


Figure 11.4 PIPER Block Read with Bus Acquisition (without byte-assembly)

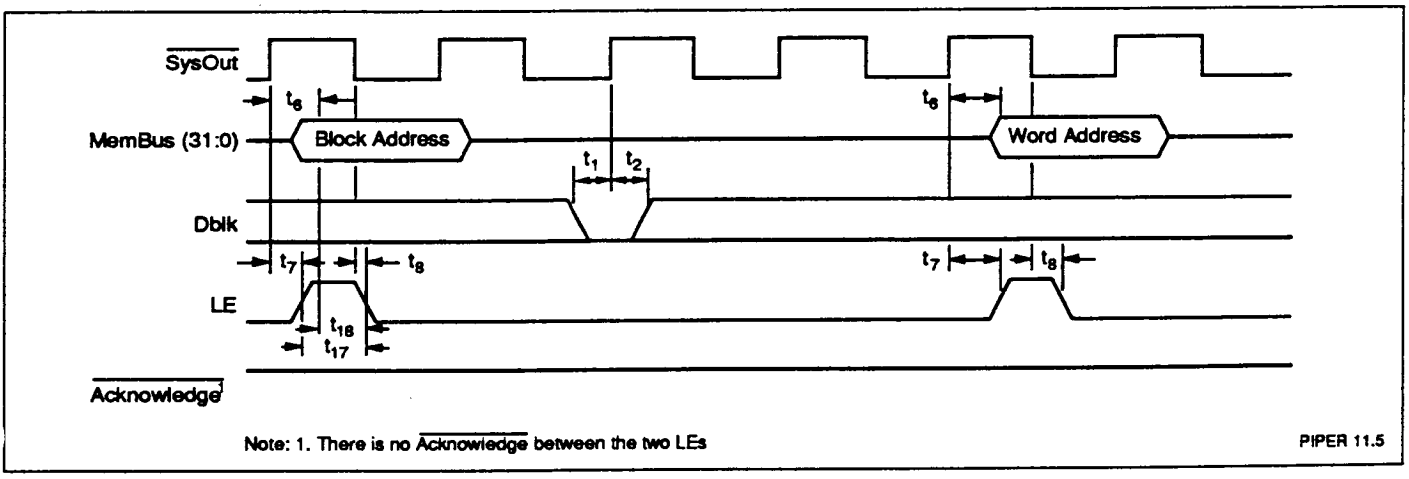


Figure 11.5 PIPER Dblk Timing

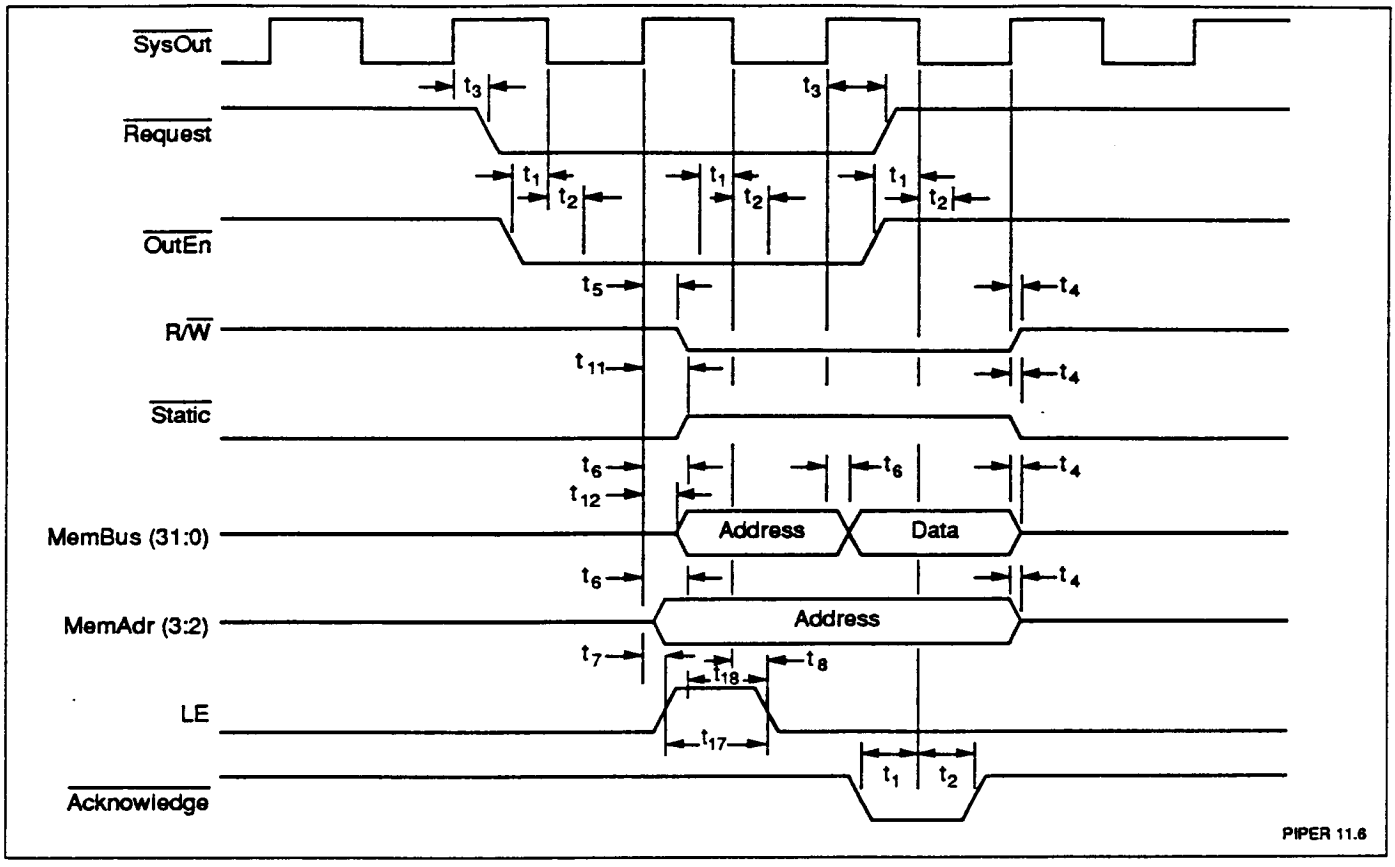
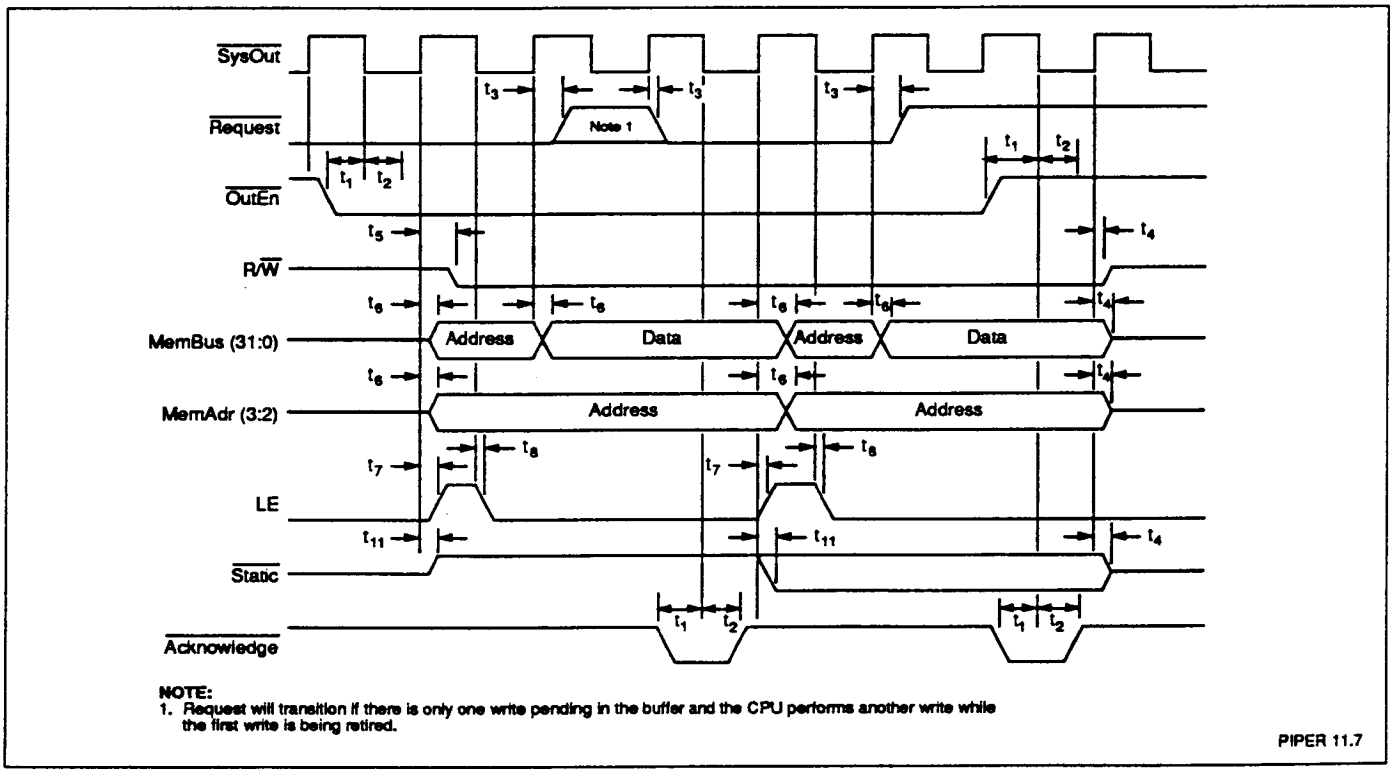


Figure 11.6 PIPER Single-Word Write with Bus Acquisition



NOTE:
1. Request will transition if there is only one write pending in the buffer and the CPU performs another write while the first write is being retired.

Figure 11.7 PIPER Multi-Word Back-to-Back Writes

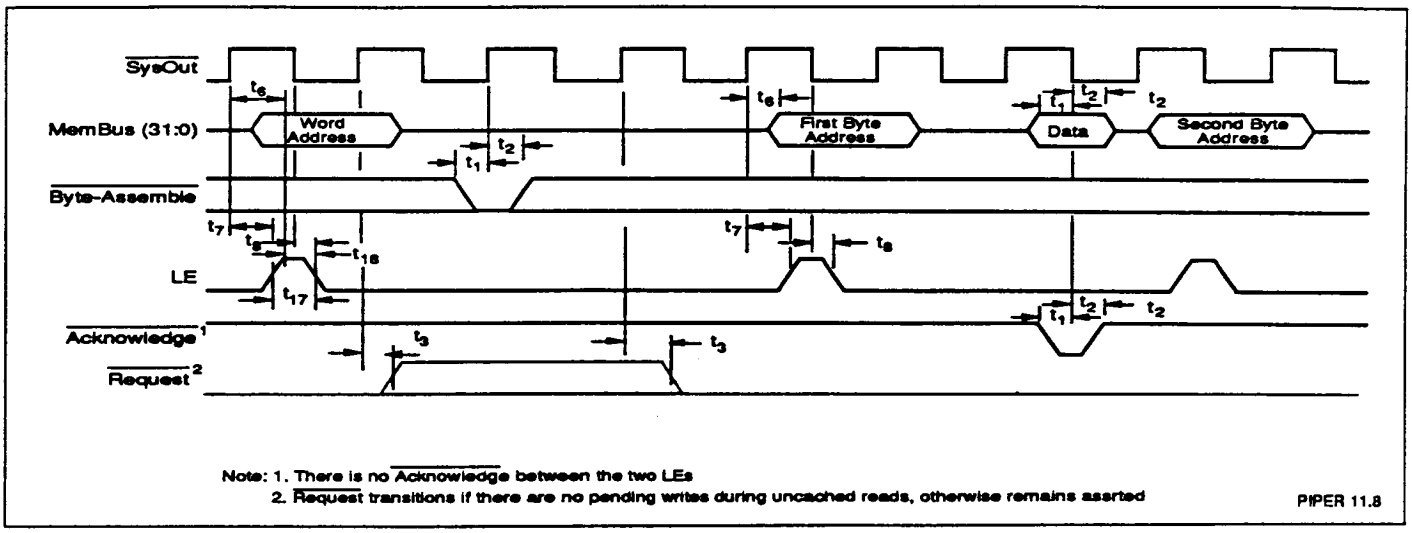


Figure 11.8 PIPER Byte-Assembly Timing

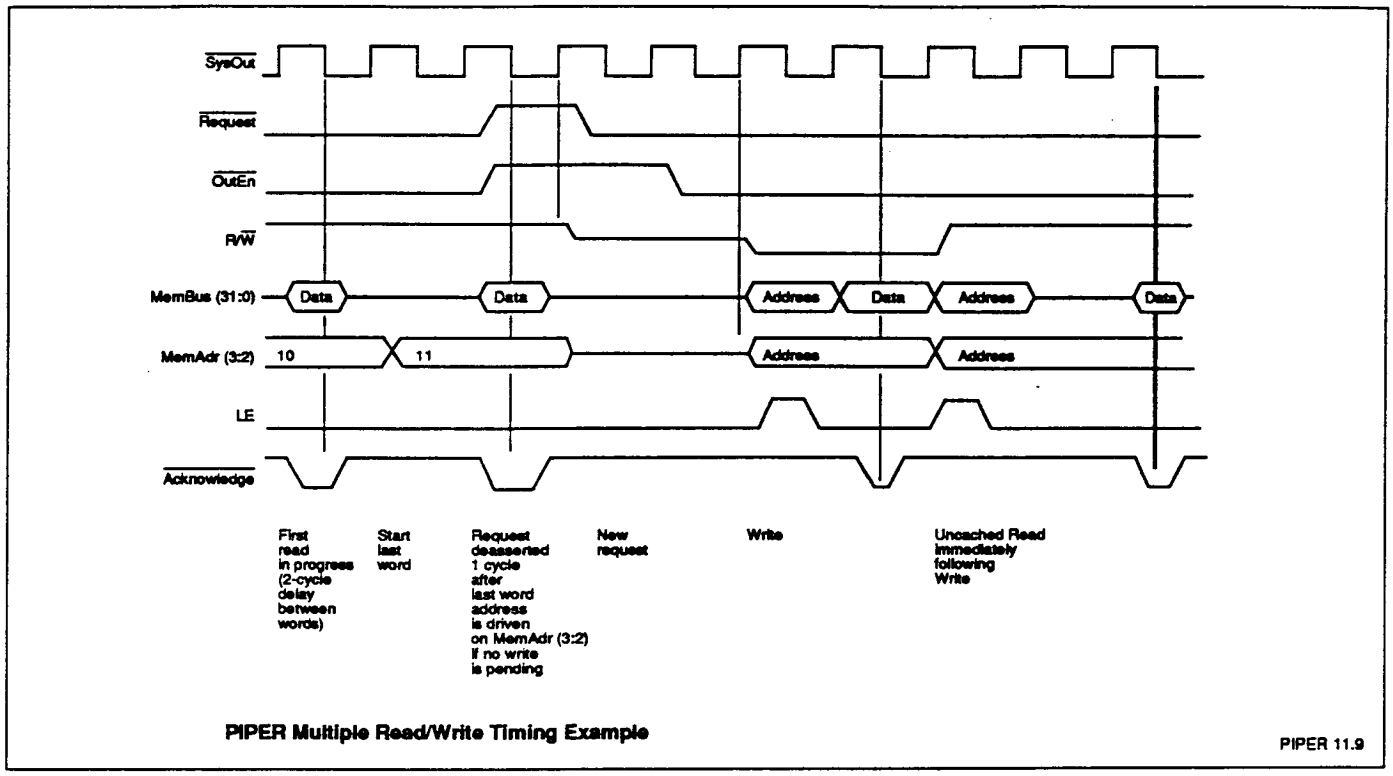


Figure 11.9 PIPER Multiple Read/Write Timing Example (Part 1)

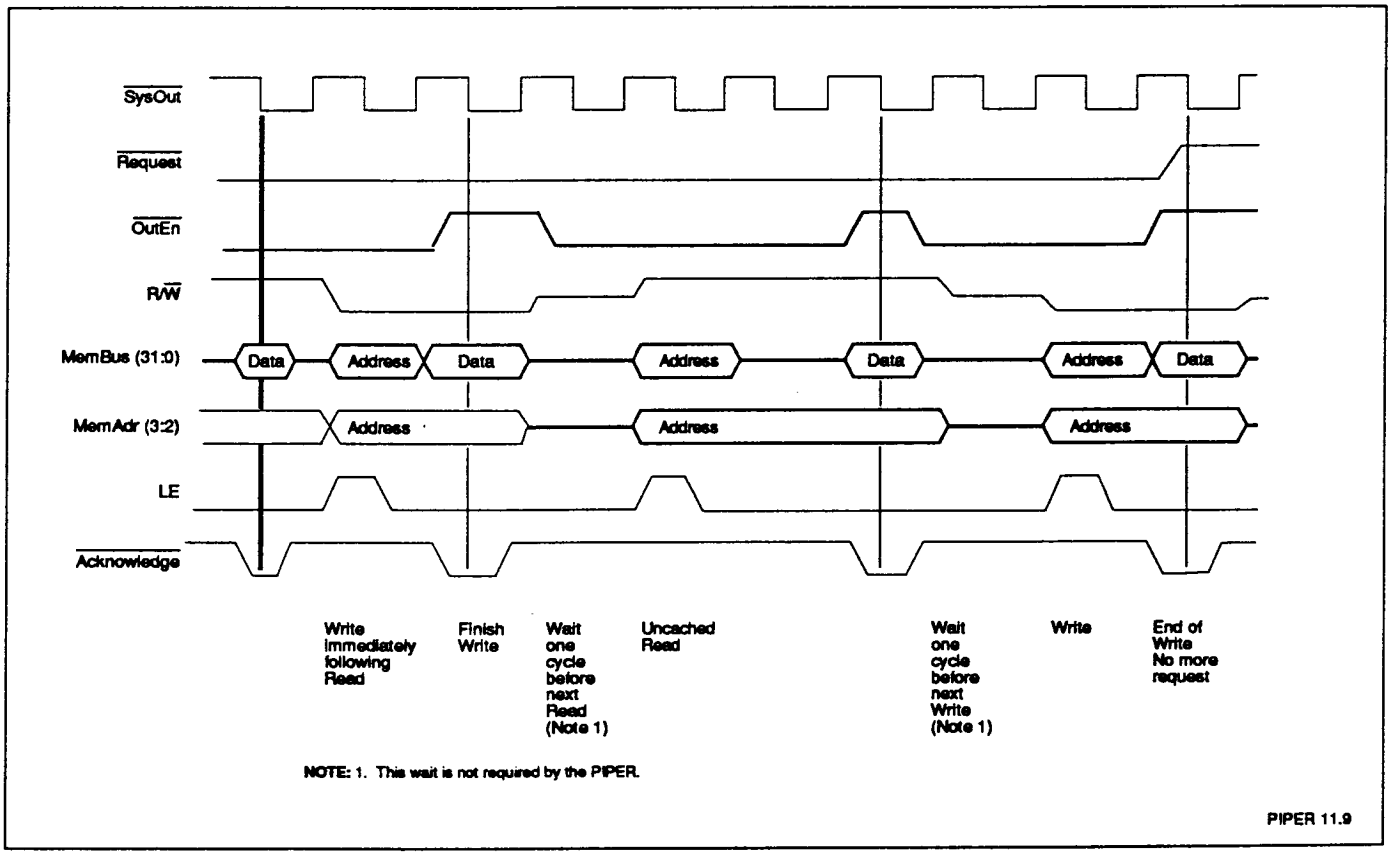


Figure 11.9 PIPER Multiple Read/Write Timing Example (Part 2)

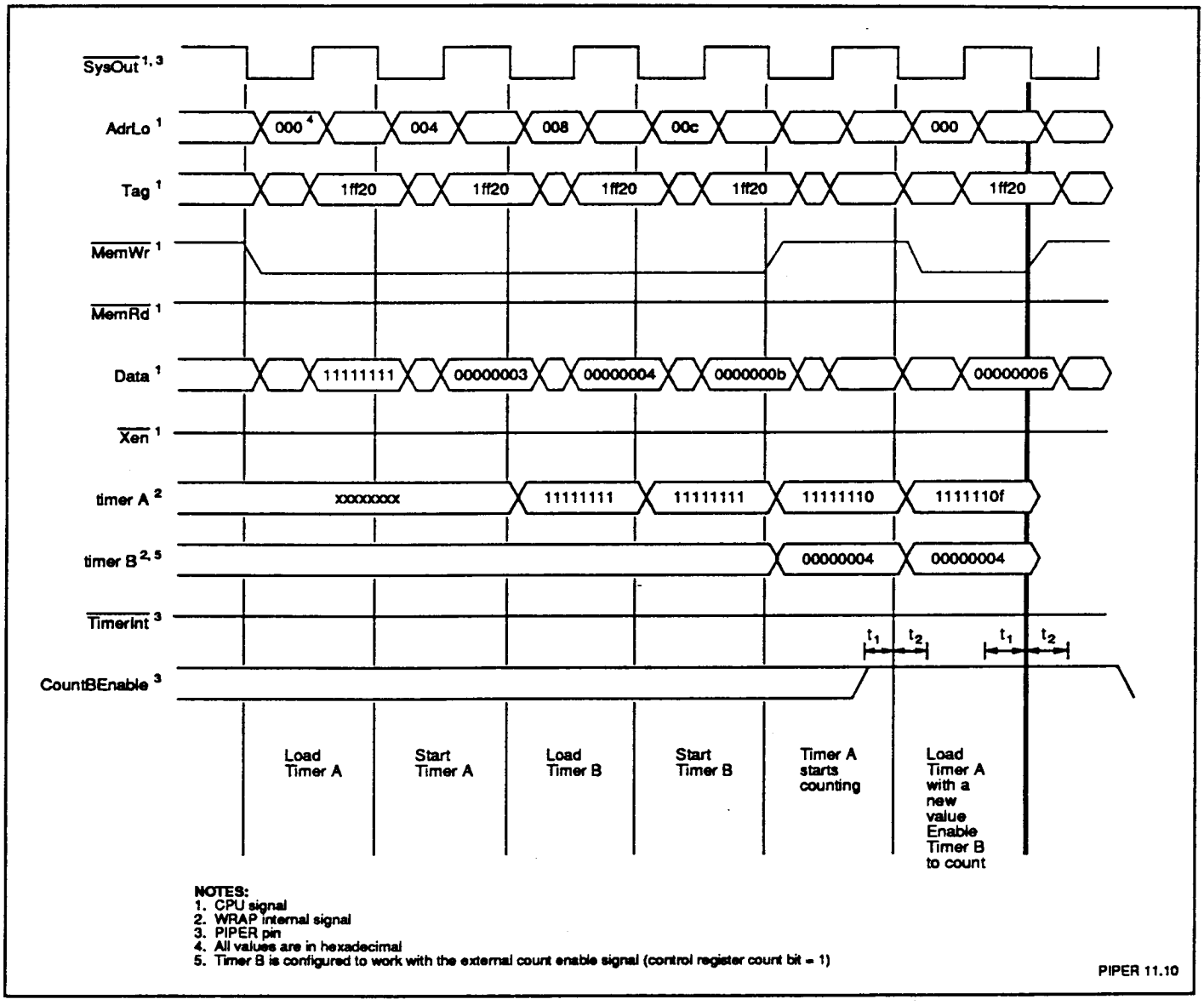


Figure 11.10 PIPER Timer/Counter Timing Example (Part 1)

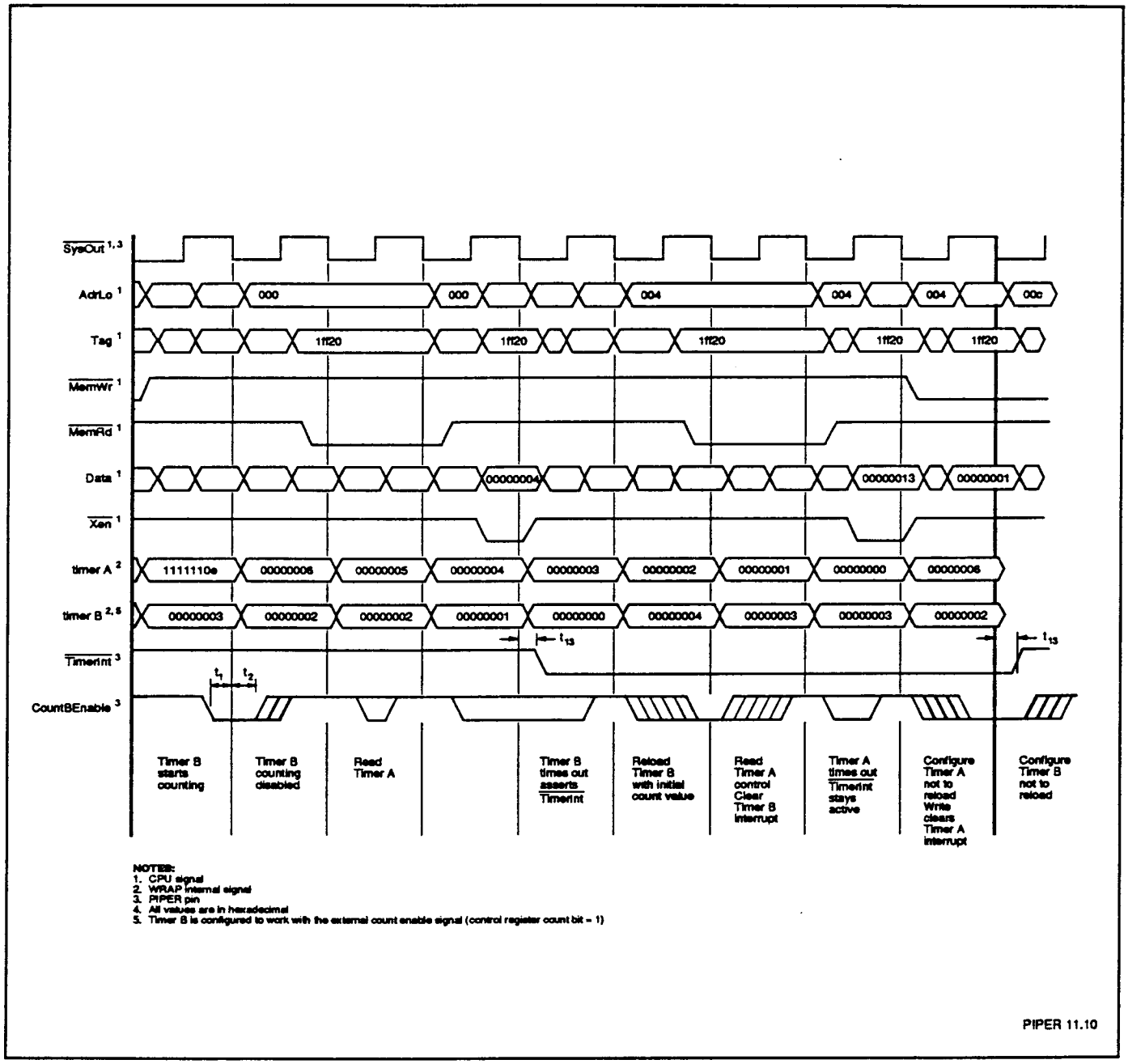
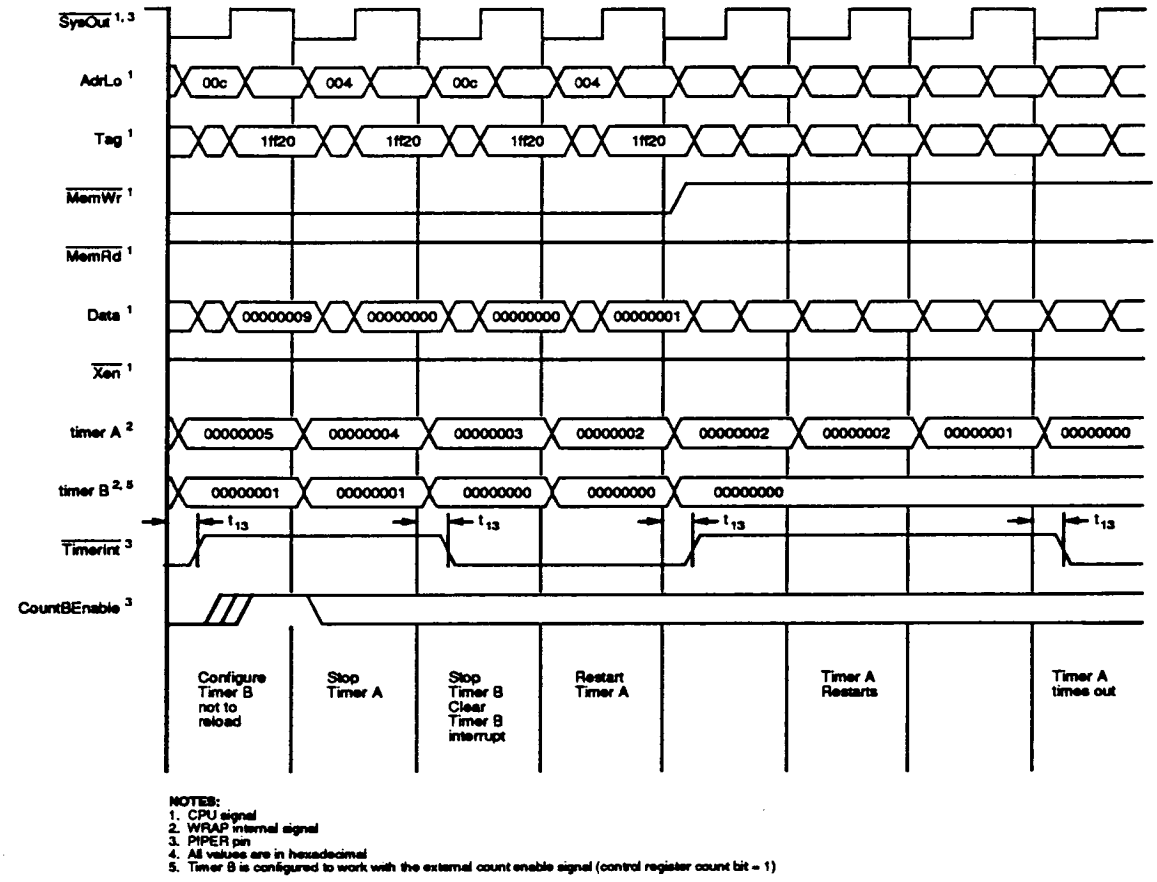


Figure 11.10 PIPER Timer/Counter Timing Example (Part 2)



NOTES:
1. CPU signal
2. WRAP internal signal
3. PIPER pin
4. All values are in hexadecimal
5. Timer B is configured to work with the external count enable signal (control register count bit = 1)

PIPER 11.10

Figure 11.10 PIPER Timer/Counter Timing Example (Part 3)

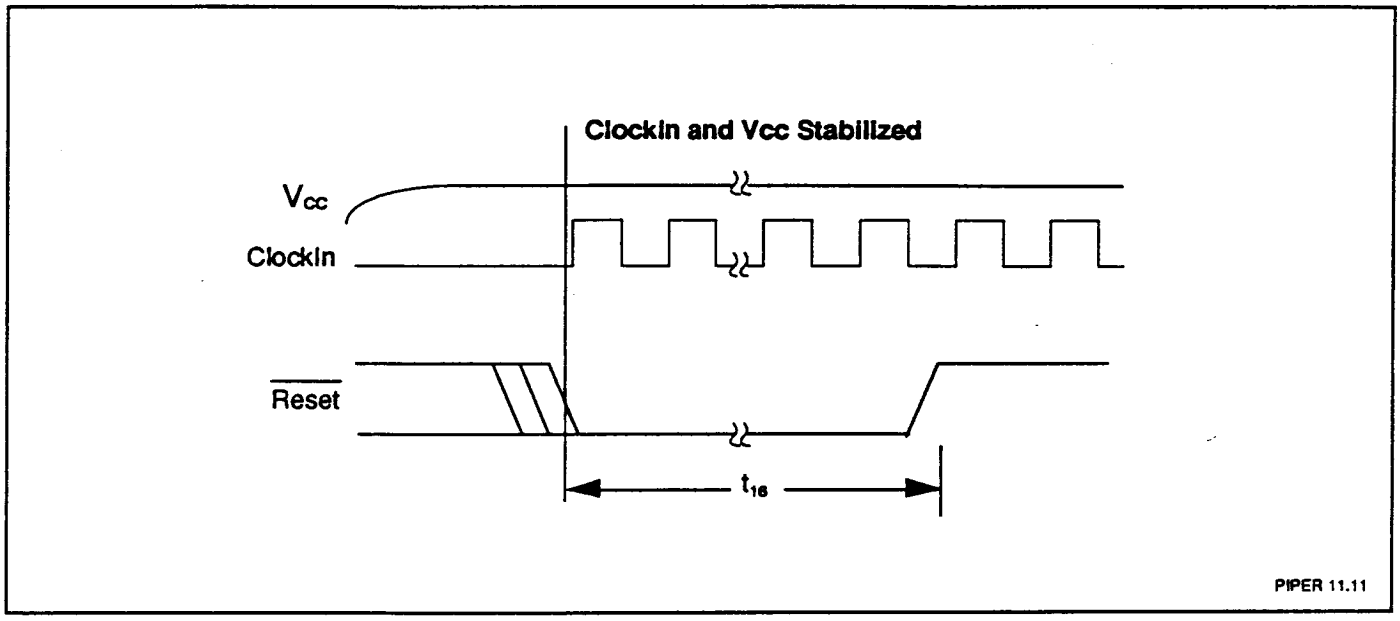


Figure 11.11 PIPER Power-On Reset Timing

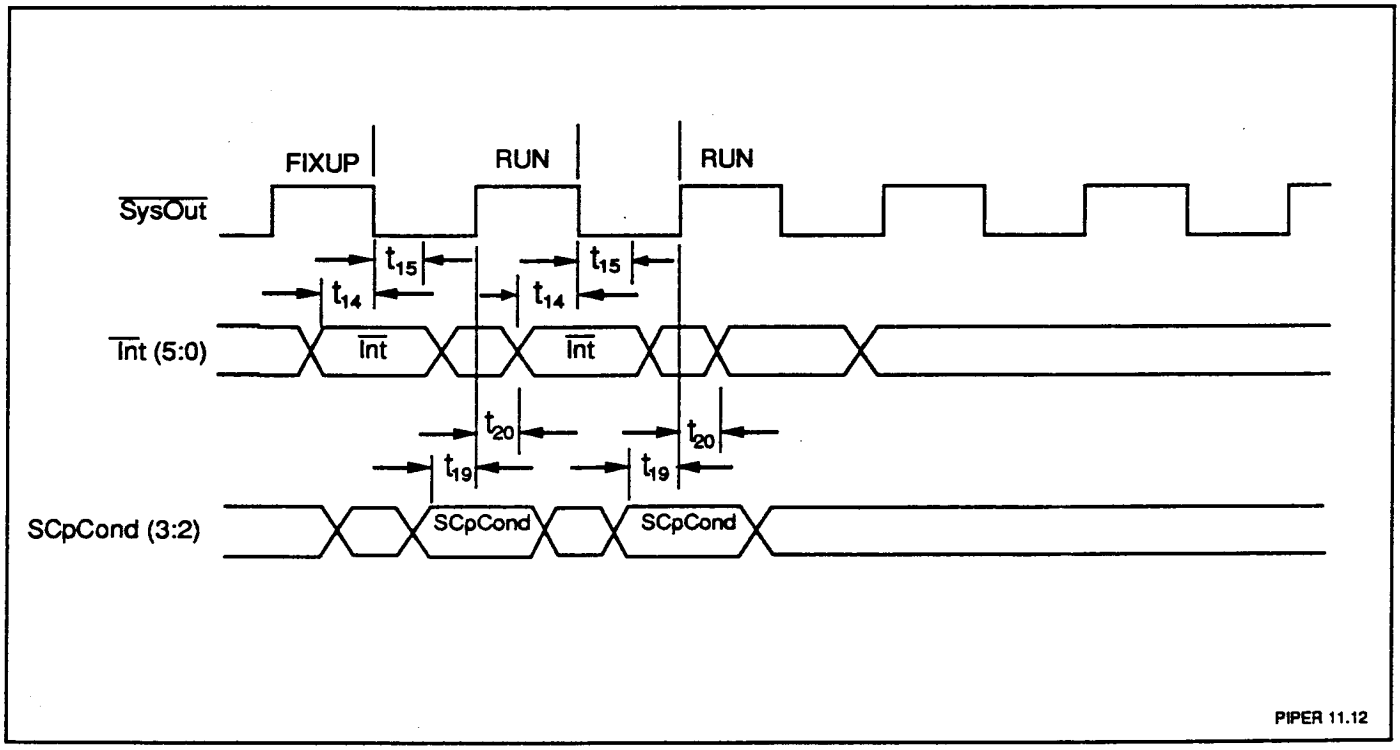


Figure 11.12 PIPER $\overline{\text{Int}}$ and SCpCond Timing

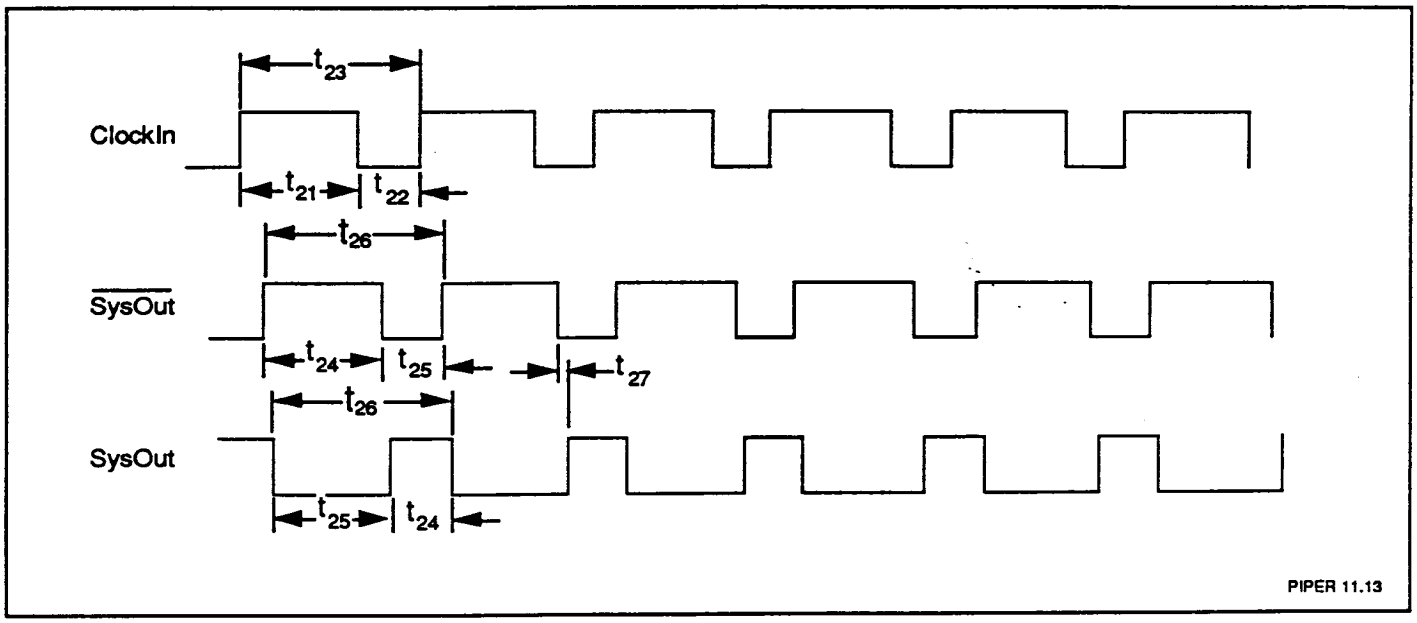


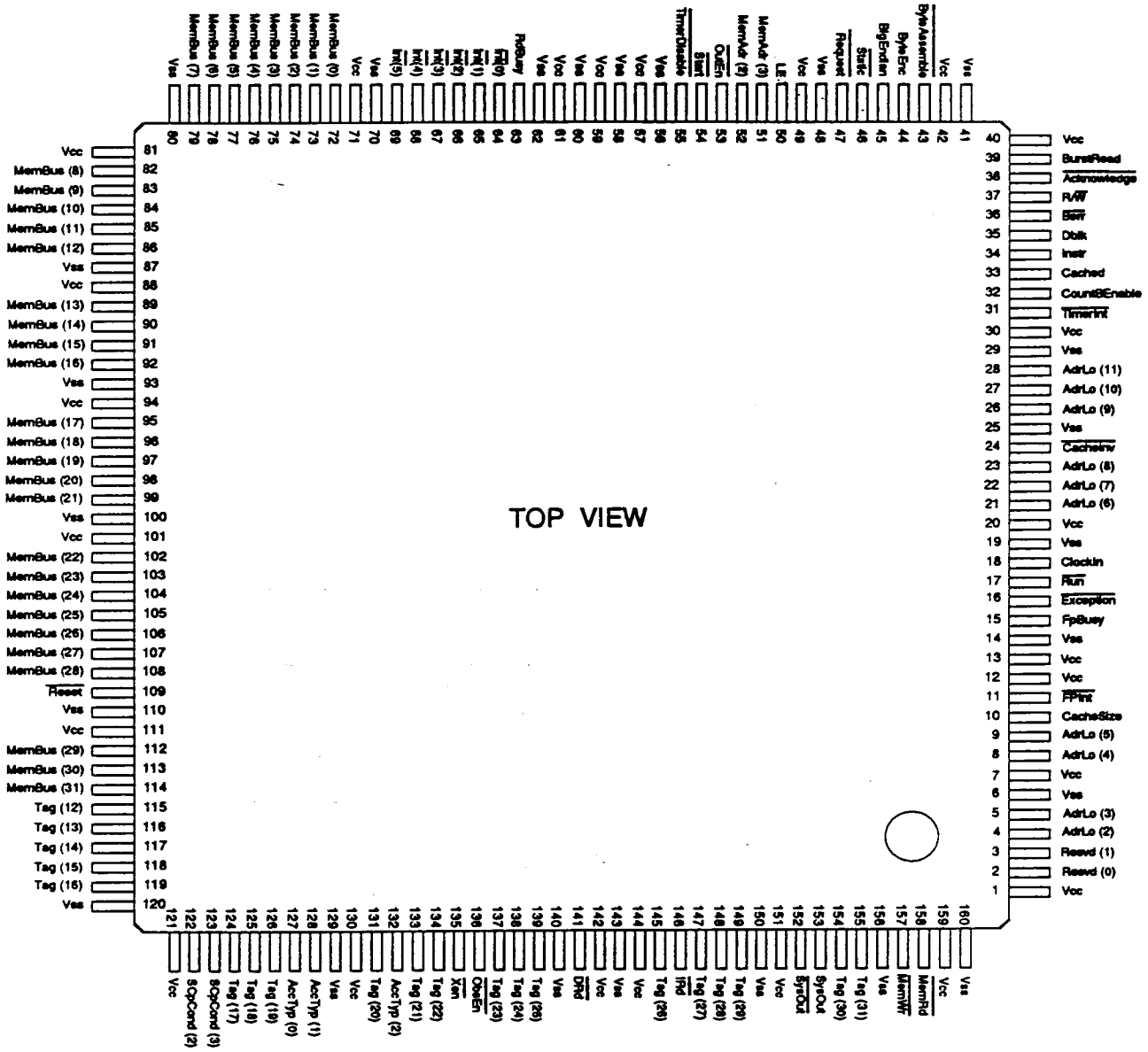
Figure 11.13 PIPER Clocks

12.0 MECHANICAL DATA - PIN ASSIGNMENTS AND PACKAGE DIMENSIONS

Table 12.1 Pinout — 160-Pin Metal Quad Flat Pack

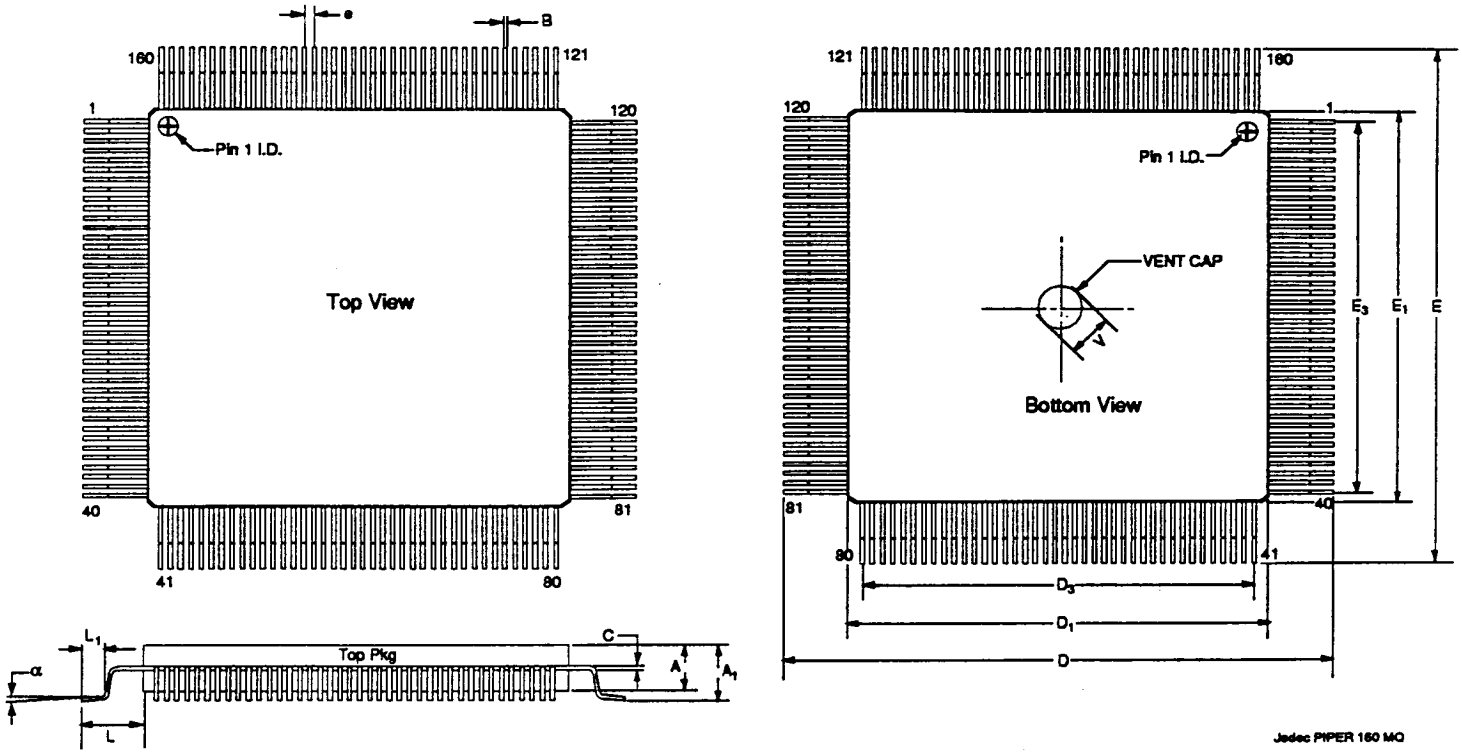
Pin No.	Pin Name	Type	Pin No.	Pin Name	Type	Pin No.	Pin Name	Type
1	VCC	VCC	55	TimerDisable	I	109	Reset	I
2	Resvd (0)	NC	56	VSS	VSS	110	VSS	VSS
3	Resvd (1)	NC	57	VCC	VCC	111	VCC	VCC
4	AdrLo(2)	O	58	VSS	VSS	112	MemBus(29)	I/O
5	AdrLo(3)	O	59	VCC	VCC	113	MemBus(30)	I/O
6	VSS	VSS	60	VSS	VSS	114	MemBus(31)	I/O
7	VCC	VCC	61	VCC	VCC	115	Tag(12)	O
8	AdrLo(4)	O	62	VSS	VSS	116	Tag(13)	O
9	AdrLo(5)	O	63	RdBusy	O	117	Tag(14)	O
10	CacheSize	I	64	Int (0)	I	118	Tag(15)	O
11	FPInt	O	65	Int (1)	I	119	Tag(16)	O
12	VCC	VCC	66	Int (2)	I	120	VSS	VSS
13	VCC	VCC	67	Int (3)	I	121	VCC	VCC
14	VSS	VSS	68	Int (4)	I	122	SCpCond(2)	I
15	FpBusy	O	69	Int (5)	I	123	SCpCond(3)	I
16	Exception	O	70	VSS	VSS	124	Tag(17)	O
17	Run	O	71	VCC	VCC	125	Tag(18)	O
18	ClockIn	I	72	MemBus(0)	I/O	126	Tag(19)	O
19	VSS	VSS	73	MemBus(1)	I/O	127	AccTyp(0)	O
20	VCC	VCC	74	MemBus(2)	I/O	128	AccTyp(1)	O
21	AdrLo(6)	O	75	MemBus(3)	I/O	129	VSS	VSS
22	AdrLo(7)	O	76	MemBus(4)	I/O	130	VCC	VCC
23	AdrLo(8)	O	77	MemBus(5)	I/O	131	Tag(20)	O
24	CacheInv	I	78	MemBus(6)	I/O	132	AccTyp(2)	O
25	VSS	VSS	79	MemBus(7)	I/O	133	Tag(21)	O
26	AdrLo(9)	O	80	VSS	VSS	134	Tag(22)	O
27	AdrLo(10)	O	81	VCC	VCC	135	Xen	O
28	AdrLo(11)	O	82	MemBus(8)	I/O	136	ObsEn	I
29	VSS	VSS	83	MemBus(9)	I/O	137	Tag(23)	O
30	VCC	VCC	84	MemBus(10)	I/O	138	Tag(24)	O
31	TimerInt	O	85	MemBus(11)	I/O	139	Tag(25)	O
32	CountBEnable	I	86	MemBus(12)	I/O	140	VSS	VSS
33	Cached	O	87	VSS	VSS	141	DRd	O
34	Instr	O	88	VCC	VCC	142	VCC	VCC
35	Dblk	I	89	MemBus(13)	I/O	143	VSS	VSS
36	Berr	I	90	MemBus(14)	I/O	144	VCC	VCC
37	R/W	O	91	MemBus(15)	I/O	145	Tag(26)	O
38	Acknowledge	I	92	MemBus(16)	I/O	146	IRd	O
39	BurstRead	I	93	VSS	VSS	147	Tag(27)	O
40	VCC	VCC	94	VCC	VCC	148	Tag(28)	O
41	VSS	VSS	95	MemBus(17)	I/O	149	Tag(29)	O
42	VCC	VCC	96	MemBus(18)	I/O	150	VSS	VSS
43	ByteAssemble	I	97	MemBus(19)	I/O	151	VCC	VCC
44	ByteEnc	I	98	MemBus(20)	I/O	152	Sysout	O
45	BigEndian	I	99	MemBus(21)	I/O	153	Sysout	O
46	Static	O	100	VSS	VSS	154	Tag(30)	O
47	Request	O	101	VCC	VCC	155	Tag(31)	O
48	VSS	VSS	102	MemBus(22)	I/O	156	VSS	VSS
49	VCC	VCC	103	MemBus(23)	I/O	157	MemWr	O
50	LE	O	104	MemBus(24)	I/O	158	MemRd	O
51	MemAdr (3)	O	105	MemBus(25)	I/O	159	VCC	VCC
52	MemAdr (2)	O	106	MemBus(26)	I/O	160	VSS	VSS
53	OutEn	I	107	MemBus (27)	I/O			
54	Start	I	108	MemBus(28)	I/O			

12.2 PIPER Pin Diagram - 160-Pin Metal Quad Flat Pack, Cavity Down, EIAJ Standard



PIPER 160 Pinout

12.3 PIPER Package Drawing - 160-Pin Metal Quad Flat Pack, Cavity Down, EIAJ Standard



Symbol	Min.		Max.	
	In.	mm.	In.	mm.
A	0.125	3,15	0.135	3,45
A_1	0.135	3,50	0.150	3,85
C	0.004	0,10	0.008	0,20
B	0.010	0,25	0.012	0,30
D/E	1.240	31,50	1,260	32,0
D_1/E_1	1.085	27,55	1.100	27,75
D_3/E_3	1.000	25,4	1.000	25,4
e	0.025 BSC	0,64 BSC	0.25 BSC	0,64 BSC
L	0.040	1,00	0.080	2,00
L_1	0.025	0,65	0.040	0,95
V	0.120	3,10	0.125	3,20
α	0°–5°	0°–5°	0°–5°	0°–5°

BSC = Basic Spacing between Centers

13.0 MOUNTING

A variety of sockets allow low insertion force or zero insertion force mountings, and a choice of terminals such as soldertail, surface mount or wire wrap. Several sockets are available from the following sample list of socket manufactures. Contact the manufacturer directly for the latest socket specifications.

- AMP Incorporated
  P.O. Box 3608
  Harrisburg, PA 17105-3608
  (800) 522-6752

- Burndy Corporation
  Richards Avenue
  Norwalk, CT 06856
  (203) 838-4444

- Yamaichi Electronics Inc.
  1425 Koll Circle, Suite 106
  San Jose, CA 95112
  (408) 452-0797

- Textool/3M Test and
  Interconnect Products Department
  3M Austin Center
  P.O. Box 2963
  Austin, TX 78769-2963
  (800) 225-5373

14.0 ORDERING INFORMATION

PACEMIPS Microprocessor Products